UNIVERSITATEA “TRANSILVANIA” DIN BRAŞOV

Laurenţiu-Mihail IVANOVICI

Procesarea Imaginilor



Îndrumar de laborator

2006

Adresa: 500030 Brasov,
B-dul Eroilor, Nr. 9
Tel/Fax: 0268 - 47 53 48
E-mail: editura@unitbv.ro



Tipărit la:
Tipografia Universităţii "Transilvania" din Braşov
B-dul Eroilor 9
tel/fax: 0268 47 53 48

Editură acreditată de CNCSIS
Adresa nr. 1615 din 29 mai 2002

Referenţi: Prof. dr. ing. Iuliu Szekely
Ş. l. dr. ing. Angel Caţaron

Descrierea CIP a Bibliotecii Naţionale a României
IVANOVICI, LAURENŢIU MIHAIL
Procesarea imaginilor: îndrumar de laborator /
Laurenţiu-Mihail Ivanovici. - Braşov: Editura Universităţii
"Transilvania", 2006
Bibliogr.
ISBN (10) 973-635-674-4; ISBN (13) 978-973-635-674-2

004.932

Cuprins

Cuprins **i**

Mulţumiri **v**

Cuvânt înainte **ix**

1 Noţiuni introductive de Qt **1**
1.1 Clasa `QApplication` 1
1.2 Clasa `QImage` 1
1.2.1 Manipularea imaginilor 2
1.2.2 Atribute ale imaginii 2
1.2.3 Manipularea pixelilor 3
1.2.4 Formate de imagine 4

2 Noţiuni introductive **5**
2.1 Percepţia imaginilor 5
2.1.1 Structura ochiului uman. Formarea imaginii 5
2.1.2 Lumina. Luminanţa. Strălucirea 6
2.2 Modelul matematic al imaginii 7
2.2.1 Modelul continuu al imaginii 7
2.2.2 Modelul discret al imaginii 8
2.3 Eşantionarea imaginilor 8
2.4 Cuantizarea imaginilor 9
2.5 Imaginile digitale 9

3 Îmbunătăţirea imaginilor **15**
3.1 Operaţiile punctuale 15
3.2 Accentuarea contrastului 16
3.3 Întinderea maximă a contrastului 17
3.4 Binarizarea imaginilor 18
3.5 Negativarea imaginilor 19
3.6 Decuparea imaginilor 20

4 **Egalizarea histogramei** **25**
4.1 Histograma unei imagini 25
4.2 Egalizarea histogramei 26
4.2.1 Algoritmul de egalizare a histogramei 26
4.3 Observaţii 27

5 **Transformări geometrice de bază** **31**
5.1 Translaţia 31
5.2 Rotaţia 31
5.3 Oglindirea 33

6 **Zgomotul în imagini** **37**
6.1 Zgomotul cu distribuţie uniformă 38
6.2 Zgomotul cu distribuţie gaussiană 39
6.3 Zgomotul de tip "sare şi piper" 41
6.4 Alte tipuri de zgomot 41

7 **Filtrarea imaginilor** **47**
7.1 Filtrarea liniară a imaginilor 47
7.1.1 Filtrele de netezire 48
7.1.2 Filtrele trece-sus 51
7.2 Filtrarea neliniară a imaginilor 52
7.2.1 Filtrele de ordine 53

8 **Transformări unitare** **59**
8.1 Transformări unitare bidimensionale 60
8.1.1 Proprietăţile transformărilor unitare 62
8.2 Transformata Fourier discretă 62
8.2.1 Transformata Fourier unidimensională 62
8.2.2 Transformarea Fourier bidimensională 63
8.3 Transformata cosinus discretă 64
8.4 Transformata sinus discretă 65

9 **Compresia imaginilor** **69**
9.1 Clasificarea metodelor de compresie 69
9.2 Algoritmul Huffman 70
9.3 Algoritmul RLE 73
9.3.1 Algoritmul RLE pentru imagini binare 73
9.3.2 Algoritmul RLE pentru imagini în tonuri de gri 73
9.4 Compresia cu transformate 74

10 **Segmentarea imaginilor** **81**
10.1 Segmentarea orientată pe regiuni 81
10.1.1 Prăguirea histogramei 82
10.1.2 Segmentarea prin creştere de regiuni 83

10.2 Segmentarea orientată pe contururi 83
10.2.1 Tehnicile de gradient 84
10.2.2 Operatorii compas 86
10.2.3 Identificarea trecerilor prin zero ale celei de-a doua derivate 87

Lista figurilor **90**

Bibliografie **92**

Dedicaţie

Doresc să dedic această carte doamnei Delia Gârbacea.

Mulţumiri

Ţin să mulţumesc domnului profesor Iuliu Szekely, conducătorul catedrei de Electronică şi Calculatoare, din cadrul Universităţii "Transilvania" din Braşov, pentru încrederea pe care mi-a acordat-o prin încredinţarea orelor de laborator de procesarea imaginilor.

Aş vrea să mulţumesc domnului profesor Vasile Buzuloiu, conducătorul Laboratorului de Analiza şi Prelucrarea Imaginilor (LAP), din cadrul Universităţii "POLITEHNICA" din Bucureşti, pentru sprijinul şi îndrumarea acordate în formarea mea profesională.

Vreau să mulţumesc colegilor din cadrul LAPI, conferenţiar Mihai Ciuc şi conferenţiar Constantin Vertan, pentru discuţiile purtate de-a lungul anilor, în special cele care au vizat lucrările de laborator de procesarea imaginilor.

Nu în ultimul rând, vreau să mulţumesc studenţilor care, cu sugestii sau comentarii, au contribuit la îmbunătăţirea conţinutului acestui îndrumar de laborator.

Cuvânt înainte

Procesarea imaginilor este un domeniu relativ recent, care evoluează rapid. Aplicaţiile sale se întâlnesc pretutindeni: în medicină, armată, industrie, artă sau acolo unde informaţia din mediul înconjurător este reprezentată sub formă de imagini. Principala aplicaţie o reprezintă îmbunătăţirea informaţiei conţinute de imagini în vederea interpretării de către un subiect uman sau pentru vederea artificială a roboţilor.

Odată cu răspândirea aparatelor de fotografiat digitale, procesarea imaginilor nu mai este domeniul exclusiv al oamenilor de ştiinţă sau al inginerilor. Aplicaţiile de procesare a imaginilor au pătruns în casa oricărui fotograf amator, fiind utilizate pentru retuşarea fotografiilor digitale.

Îndrumarul de faţă se adresează în special studenţilor de la facultăţile tehnice, care urmează cursul de Procesarea imaginilor, dar şi celor care doresc să îşi însuşească algoritmii fundamentali de procesare a imaginilor. Lucrările de laborator prezentate abordează operaţiile punctuale, de vecinătate şi cele integrale, de la accentuarea contrastului până la transformări unitare. Ultimul capitol, referitor la segmentarea imaginilor, reprezintă o incursiune în domeniul analizei imaginilor.

Scopul lucrărilor de laborator este acela de a prezenta noţiunile fundamentale de procesare a imaginilor alb-negru şi de a exemplifica folosind limbajul C++ modalităţi de implementare a algoritmilor de procesare prezentaţi. Pentru o înţelegere cât mai bună a soluţiilor propuse, cititorul trebuie să fie familiarizat cu limbajele de programare C şi C++.

Lucrarea 1

Noţiuni introductive de Qt

Qt[1] este un mediu de dezvoltare care include biblioteci de clase C++ şi unelte de dezvoltare de aplicaţii pentru diverse sisteme de operare: Microsoft Windows, MAC OS sau Linux.

Bibliotecile **Qt** cuprind peste 400 de clase C++, care încapsulează infrastructura necesară pentru dezvoltarea de aplicaţii. **Qt** API[2] include clase pentru realizarea de interfeţe grafice utilizator, pentru programarea în reţea, baze de date sau integrare OpenGL[3].

1.1 Clasa `QApplication`

Clasa `QApplication` controlează şi gestionează interfaţa grafică a unei aplicaţii. Conţine bucla principală de evenimente a aplicaţiei, în care sunt procesate evenimentele provenind de la sistemul de ferestre. De asemenea gestionează configurările aplicaţiei, fazele de iniţializare şi terminare a aplicaţiei.

Orice aplicaţie **Qt** este un obiect `QApplication` indiferent de numărul de ferestre ale aplicaţiei. Obiectul este accesibil prin apelarea funcţiei membre `instance()`, care returnează un pointer către acest obiect.

1.2 Clasa `QImage`

Clasa `QImage` pune la dispoziţie o reprezentare independentă de platformă a unei imagini, care permite accesul direct la valorile pixelilor. **Qt** oferă patru clase pentru manipularea imaginilor: `QImage`, `QPixmap`, `QBitmap` şi `QPicture`.

Clasa `QImage` este proiectată şi optimizată pentru operaţii de intrare/ieşire şi pentru acces direct la pixelii imaginii, pe când clasa `QPixmap` este proiectată şi optimizată pentru vizualizarea imaginilor pe ecran. Clasa `QBitmap`

[1] Trolltech, http://www.trolltech.com
[2] Application Program Interface.
[3] Limbaj de grafică 3D dezvoltat de firma Silicon Graphics.

moşteneşte clasa `Pixmap` şi este folosită pentru imagini binare, iar clasa `QPicture` este un "paint device". În continuare ne vom concentra atenţia asupra clasei `QImage`.

Clasa `QImage` poate manipula o serie de formate de imagine, care includ fie imagini monocrome, fie reprezentate pe 8 biţi sau 32 biţi. Clasa pune la dispoziţie o colecţie de funcţii care pot fi folosite pentru obţinerea de informaţii despre imagine sau care permit anumite transformări ale imaginii.

1.2.1 Manipularea imaginilor

Clasa `QImage` oferă câteva moduri de a încărca o imagine dintr-un fişier: la instanţierea obiectului QImage sau folosind funcţiile `loadFromData()` sau `load()`, care vor fi apelate după crearea obiectului `QImage`. Pentru a salva un obiect de tip `QImage` se foloseşte funcţia `save()`.

Următorul exemplu ilustrează folosirea funcţiei `load()`:

```
QImage image;
QString filename = QFileDialog :: getOpenFileName();
image.load( filename, 0 );
```

Lista completă a formatelor de fişiere cunoscute este disponibilă prin intermediul a două metode: `QImageReader::supportedImageFormats()` şi `QImageWriter::supportedImageFormats()`. Implicit, **Qt** suportă următoarele formate de fişiere:

Format	**Descriere**	**Operaţii**
BMP	Windows Bitmap	Read/Write
GIF	Graphic Interchange Format (opţional)	Read
JPG	Joint Photographic Experts Group	Read/Write
JPEG	Joint Photographic Experts Group	Read/Write
PNG	Portable Network Graphics	Read/Write
PBM	Portable Bitmap	Read
PGM	Portable Graymap	Read
PPM	Portable Pixmap	Read/Write
XBM	X11 Bitmap	Read/Write
XPM	X11 Pixmap	Read/Write

Tabelul 1.1: Formate de fişiere imagine suportate de **Qt**.

1.2.2 Atribute ale imaginii

`QImage` oferă o colecţie de funcţii pentru obţinerea de informaţii despre atributele imaginii, cum ar fi geometria imaginii sau informaţii despre paleta de culori.

Atribute	Funcţii
Geometrie	Funcţiile `size()`, `width()`, `height()`, `dotsPerMeterX()` şi `dotsPerMeterY()` oferă informaţii despre dimensiunile şi aspectul imaginii.
Culoare	Culoarea unui pixel poate fi aflată specificând coordonatele pixelului funcţiei `pixel()`, care returnează culoarea ca o valoare `QRbg`. În cazul imaginilor monocrome sau cu nivele de gri, funcţiile `numColors()` şi `colorTable` oferă informaţii despre componentele de culoare ale imaginii.
Nivel jos	Funcţia `depth()` returnează numărul de biţi pe care este reprezentată valoarea unui pixel: 1 (imagini monocrome), 8 sau 32. Funcţiile `format()`, `bytesPerLine()` şi `numBytes()` oferă informaţii despre modul de stocare a imaginii.

Tabelul 1.2: Funcţii de aflare a atributelor unei imagini.

1.2.3 Manipularea pixelilor

În cazul în care culorile pixelilor sunt stocate pe 32 biţi, funcţia `setPixel` se poate utiliza pentru modificarea valorii unui pixel specificat prin coordonatele sale. Noua sa valoare este un cvadruplu ARGB[4], fiind unul din argumentele funcţiei. Pentru a specifica valoarea unui pixel în modelul RGB se foloseşte funcţia `qRgb` care returnează un obiect QRgb.

În continuare este prezentat un exemplu de folosire a funcţiei `setPixel`. Se crează mai întâi un obiect `QImage`, reprezentând o imagine de dimensiune 3 x 3.

```
QImage image(3, 3, QImage::Format_RGB32);
QRgb value;

value = qRgb(189, 149, 39); // 0xffbd9527
image.setPixel(1, 1, value);

value = qRgb(122, 163, 39); // 0xff7aa327
image.setPixel(0, 1, value);
image.setPixel(1, 0, value);
```

[4]Alpha Red Green Blue.

```
value = qRgb(237, 187, 51); // 0xffedba31
image.setPixel(2, 1, value);
```

1.2.4 Formate de imagine

Fiecare pixel stocat într-un obiect de tip `QImage` este reprezentat ca un întreg. Dimensiunea acestui întreg depinde de format. **Qt** suportă imagini monocrome, reprezentate pe 1 bit şi imagini reprezentate pe 8 sau 32 biţi.

Imaginile monocrome sunt stocate folosind indici de 1 bit într-o tabelă de culoare cu doar două intrări (culori). În funcţie de ordinea de stocare a biţilor, big endian sau little endian, diferenţiem două tipuri de imagini monocrome.

Imaginile reprezentate pe 8 biţi sunt stocate folosind indici de 8 biţi într-o tabelă de culoare, având un byte per pixel. Tabela de culoare este un obiect `QVector<QRgb>`.

Imaginile reprezentate pe 32 biţi nu au tabelă de culoare, fiecare pixel conţinând o valoare `QRgb`. Cea mai răspândită modalitate de a reprezenta tripletul RGB pe 32 biţi este următoarea: 0xffRRGGBB, în care se folosesc câte 8 biţi pentru fiecare componentă.

Formatul unei imagini se poate determina folosind funcţia `format()`.

Pentru a converti o imagine într-un nou format se foloseşte funcţia `convertToFormat()`, care acceptă ca argument noul format al imaginii.

Lucrarea 2

Noţiuni introductive de prelucrarea imaginilor

BREVIAR TEORETIC

2.1 Percepţia imaginilor

Înţelegerea procesului de percepţie a imaginilor de către ochiul uman este foarte importantă pentru dezvoltarea de tehnici de evaluare a calităţii unui sistem sau algoritm de procesare a imaginilor. Informaţia la nivel vizual, conţinută de către o imagine, reprezintă o distribuţie spaţială a unei anumite mărimi, cum ar fi de exemplu luminanţa obiectelor ce compun respectiva imagine. Informaţia percepută de ochiul uman poate fi definită de atribute cum ar fi strălucirea, culoarea sau muchiile unui obiect.

2.1.1 Structura ochiului uman. Formarea imaginii

Ochiul uman are o formă aproape sferică, având un diametru, în medie, de aproximativ 2 cm. Este format din mai multe membrane: *corneea* şi *sclerotica*, ca înveliş exterior, şi *coroida* şi *retina* în interior. Corneea este un ţesut dur şi transparent, pe când sclerotica este o ţesut opac. Coroida se află imediat sub sclerotică şi conţine o reţea de vase de sânge, ce hrănesc ochiul. Este puternic pigmentată pentru a reduce excedentul de lumină ce intră în ochi. Pe cea mai din interior membrană, retina, se formează imaginea, sub influenţa luminii reflectate de obiectele exterioare ochiului.

Retina conţine două tipuri de receptori: *conuri* şi *bastonaşe*. Conurile, în număr de aproximativ 6-7 milioane, servesc la percepţia culorilor. Vederea umană poate percepe detalii foarte fine datorită densităţii mari a acestor receptori, fiecare con fiind legat la o terminaţie nervoasă. Bastonaşele, în număr de aproximativ 75-150 milioane, servesc vederii crepusculare, în condiţii de iluminare slabă. Acestea sunt răspândite pe o arie mare şi conec-

tate la doar câteva terminaţii nervoase, ceea ce are ca efect reducerea considerabilă a perceperii detaliilor din imagine.

Cristalinul joacă rolul de lentilă. El conţine aproximativ 70% apă şi, prin compoziţia lui, permite trecerea a doar 8% din spectrul de radiaţie vizibilă, absorbind în bună măsură şi radiaţiile infraroşii şi ultraviolete.

Imaginea obiectelor se proiectează pe retină, prin cristalin, fiind răsturnată şi având dimensiuni mult mai mici, principiu ce stă la baza aparatului de fotografiat (vezi Figura 2.1).

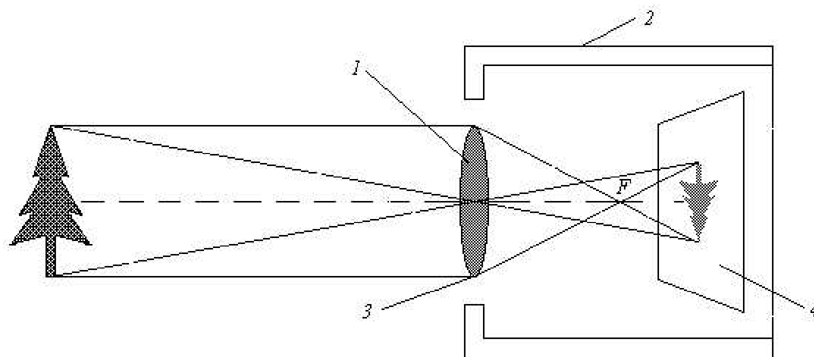


Figura 2.1: Formarea imaginii în aparatul de fotografiat: 1-lentilă, 2-camera obscură, 3-obiectiv, 4-peliculă.

2.1.2 Lumina. Luminanţa. Strălucirea

Lumina este o radiaţie electromagnetică ce stimulează receptorii de la nivelul retinei. Ea se exprimă ca fiind o distribuţie $L(\lambda)$ de energie, unde λ este lungimea de undă a radiaţiei, în cazul nostru vizibilă, cu valori între 350 şi 780 nm. Lumina percepută de la un obiect se poate scrie matematic astfel:

$$I(\lambda) = \rho(\lambda)L(\lambda) \tag{2.1}$$

unde $\rho(\lambda)$ reprezintă măsura în care un obiect reflectă sau transmite energia luminoasă incidentă, a cărei distribuţie este exprimată prin $L(\lambda)$.

Luminanţa sau intensitatea luminoasă a unui obiect cu o distribuţie spaţială a luminii, $I(x, y, \lambda)$, se defineşte astfel:

$$f(x,y) = \int_0^{\infty} I(x,y,\lambda)V(\lambda)d\lambda \tag{2.2}$$

unde $V(\lambda)$ este *funcţia de eficienţă luminoasă relativă* a sistemului vizual. Pentru ochiul uman, $V(\lambda)$ este o curbă de tip clopot, a cărei caracteristici depind de la o persoană la alta (vezi Figura 2.2).

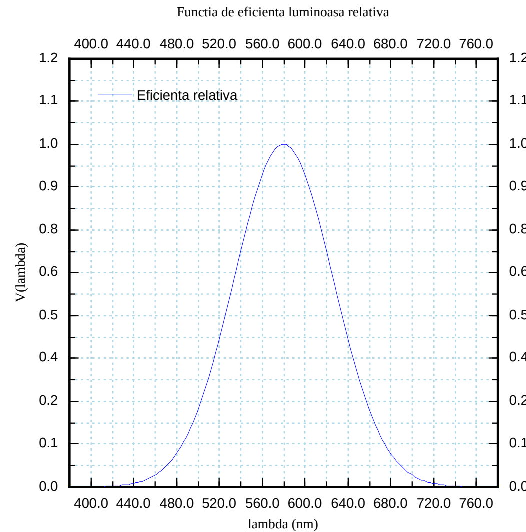


Figura 2.2: Forma tipică a funcţiei de eficienţă luminoasă relativă.

Luminanţa unui obiect este independentă de luminanţa obiectelor din jur.

Strălucirea unui obiect este luminanţa percepută şi depinde de luminanţa mediului ambiant obiectului. Două obiecte aflate într-un acelaşi ambient pot avea aceeaşi luminanţă, dar străluciri diferite.

2.2 Modelul matematic al imaginii

2.2.1 Modelul continuu al imaginii

Matematic imaginile pot fi reprezentate ca o funcţie de două variabile, în spaţiul $L^2(R^2)$, astfel:

- imaginile în tonuri de gri se pot modela cu: $f(x,y) : R^2 \rightarrow R$, caz în care valorile funcţiei f reprezintă valorile luminanţei obiectelor din imagine, în punctele (x,y) ale spaţiului.

- imaginile color se pot modela cu: $f(x,y) : R^2 \rightarrow R^3$, caz în care valorile funcţiei f reprezintă vectori de 3 componente dintr-un spaţiu al culorilor. De exemplu pot fi cele trei componente ale modelului RGB[1].

Spaţiul $L^2(R^2)$ poate fi limitat la un domeniu finit D, ca în Figura 2.3.

[1]RGB = (engl.) Red Green Blue.

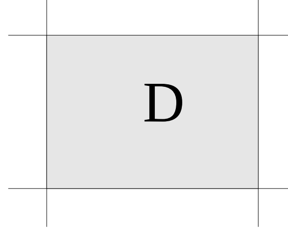


Figura 2.3: Domeniu finit din R^2.

2.2.2 Modelul discret al imaginii

Acesta este modelul utilizat în practică. Funcţia f ia valori discrete, fiind deasemenea definită pe un domeniu de valori discrete, adică:

$$f(k,l) : Z^2 \rightarrow Z^+ \ \ sau \ \ f(k,l) : Z^2 \rightarrow Z^{+3} \tag{2.3}$$

Trecerea de la domeniul continuu la domeniul discret se face prin eşantionare şi cuantizare.

2.3 Eşantionarea imaginilor

Pentru a putea prelucra cu ajutorul unui calculator o imagine $f(x,y)$, aceasta trebuie discretizată spaţial şi în amplitudine. Discretizarea coordonatelor spaţiale (x,y) poartă numele de *eşantionare*.

Eşantionarea reprezintă procesul de aproximare a unei imagini continue $f(x,y)$ cu o matrice de dimensiune MxN, astfel:

$$f(x,y) \approx \begin{pmatrix} f(0,0) & f(0,1) & ... & f(0,M-1) \\ f(1,0) & f(1,1) & ... & f(1,M-1) \\ . & . & ... & . \\ . & . & ... & . \\ f(N-1,0) & f(N-1,1) & ... & f(N-1,M-1) \end{pmatrix} \tag{2.4}$$

Teorema eşantionării: O imagine $f(x,y)$, având un spectru finit, eşantionată uniform cu o reţea dreptunghiulară de forma celei din Figura 2.4 poate fi refăcută fără eroare din eşantioanele $f(m\delta x, n\delta y)$ cu ajutorul formulei de interpolare:

$$f(x,y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(m\delta x, n\delta y) \left(\frac{sin\pi(xu_s - m)}{\pi(xu_s - m)} \right) \left(\frac{sin\pi(xv_s - n)}{\pi(xv_s - n)} \right) \tag{2.5}$$

unde u_s şi v_s reprezintă frecvenţele spaţiale de eşantionare.

Egalitatea dată de teorema eşantionării este valabilă dacă şi numai dacă este respectată condiţia Nyquist, şi anume:

$$\frac{1}{\delta x} = u_s > 2u_0, \qquad \frac{1}{\delta y} = v_s > 2v_0 \tag{2.6}$$

unde u_0 şi v_0 reprezintă frecvenţele spaţiale maxime care apar în imagine.

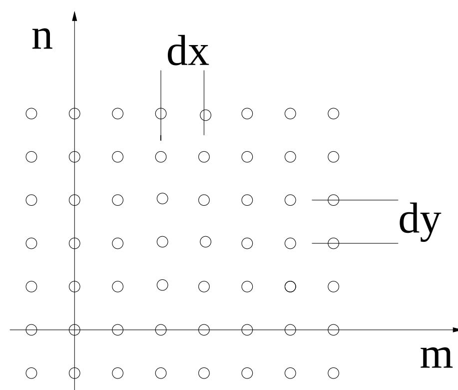


Figura 2.4: Reţea dreptunghiulară de eşantionare.

Cu alte cuvinte, frecvenţele spaţiale de eşantionare trebuie să fie cel puţin dublul frecvenţelor spaţiale maxime conţinute de imagine.

2.4 Cuantizarea imaginilor

Cuantizarea este procesul de discretizare a valorilor funcţiei $f(x, y)$. Aceasta se realizează de obicei cu ajutorul unei funcţii de tip scară, de forma celei din Figura 2.5.

Astfel, tuturor valorilor lui x dintr-un interval li se atribuite valori discrete k. Cuantizarea este un proces însoţit de zgomot, cunoscut sub numele de *eroare de cuantizare*. Cea mai utilizată metodă de cuantizare este cea uniformă, ceea ce înseamna că intervalele funcţiei de cuantizare sunt egale.

2.5 Imaginile digitale

Imaginile astfel discretizate reprezintă structuri bidimensionale de date, denumite imagini digitale. Un element (k, l) al imaginii poartă numele de *pixel*[2].

[2]pixel = (engl. picture + element).

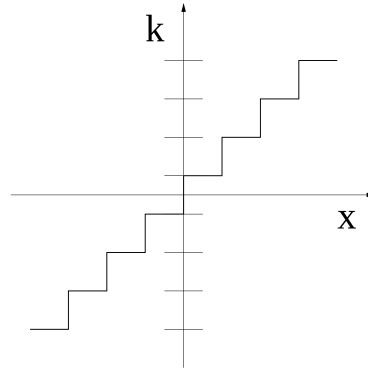


Figura 2.5: Exemplu de funţie de cuantizare.

Imaginile digitale pot fi stocate în memoria sau pe discul unui sistem de procesare şi analiză a imaginilor, în vederea vizualizării sau prelucrării ulterioare. Pe disc imaginile sunt stocate sub forma unor fişiere. Fişierele pot fi de mai multe feluri, în funcţie de formatul în care păstrează datele ce reprezintă imagini: BMP, JPEG, GIF, TIFF, etc.

Cel mai simplu format de fişier imagine este Windows Bitmap (BMP) al firmei Microsoft. Acesta este formatul în care o imagine digitală este stocată practic fără nici un fel de codare sau pierdere de informaţie, cu excepţia reprezentării binare.

DESFĂŞURAREA LUCRĂRII

În continuare este prezentat codul C++ al unei aplicaţi care citeşte o imagine în format BMP şi o afişează într-o fereastră pe ecran. Aplicaţia este dezvoltată utilizând bibliotecile Qt (Linux). Citiţi şi înţelegeţi codul.

Fişierul `aplicatie.h` conţine declararea clasei ImageViewer.

```
#ifndef APLICATIE_H
#define APLICATIE_H

#include <qwidget.h>
#include <qimage.h>
#include <qpainter.h>
#include <qlabel.h>

class QMenuBar;
class QPopupMenu;

class ImageViewer : public QWidget
```

```
{
   Q_OBJECT
   public:
      ImageViewer( QWidget *parent = 0, const char *name = 0,
                   int wFlags = 0 );
      ~ImageViewer();
   bool loadImage( const char *fileName );

   protected:
      void paintEvent( QPaintEvent * );

   private:
      int        conversion_flags;
      int        alloc_context;
      QImage     image;
      QPixmap    pm;
      QMenuBar   *menubar;
      QPopupMenu *file;
      QLabel     *status;
      bool       reconvertImage();

   private slots:
      void  openFile( void );
      void  saveFile( void );
};

#endif // APLICATIE_H
```

Fişierul `aplicatie.cpp` conţine definiţiile funcţiilor membre clasei ImageViewer.

```
#include "aplicatie.h"
#include <qmenubar.h>
#include <qfiledialog.h>
#include <qmessagebox.h>
#include <qpopupmenu.h>
#include <qpainter.h>
#include <qapplication.h>
#include <qwidget.h>

ImageViewer :: ImageViewer( QWidget *parent, const char *name,
    int wFlags ) : QWidget( parent , name , wFlags ),
    conversion_flags( PreferDither ), filename( 0 )
{
   alloc_context = 0;
```

```
   menubar = new QMenuBar( this );
   menubar->setSeparator( QMenuBar :: InWindowsStyle );

   file = new QPopupMenu();
   CHECK_PTR( file );
   menubar->insertItem( "&File" , file );
   file->insertItem( "Open image ( BMP )", this,
         SLOT( openFile() ), CTRL+Key_O );
   file->insertSeparator();
   file->insertItem( "Save as ... ( BMP )", this,
         SLOT( saveFile() ), CTRL+Key_S );
   file->insertSeparator();
   file->insertItem( "Quit", qApp, SLOT(quit()), CTRL+Key_Q );

   status = new QLabel( this );
   status->setFrameStyle( QFrame::WinPanel | QFrame::Sunken );
   status->setFixedHeight( fontMetrics().height() + 4 );

   setMouseTracking( TRUE );
}

ImageViewer :: ~ImageViewer()
{
   if ( alloc_context )
   QColor :: destroyAllocContext( alloc_context );

   delete menubar;
   delete file;
   delete help;
   delete status;
}

void ImageViewer  :: openFile()
{
   QString newfilename = QFileDialog :: getOpenFileName();

   if ( !newfilename.isEmpty() )
   {
      loadImage( newfilename );
      repaint();
   }
}

void ImageViewer :: saveFile( void )
```

```
{
   QImageIO iio;
   QString save_as_file = QFileDialog :: getSaveFileName();

   if( !save_as_file.isEmpty() )
   {
      iio.setImage( image );
      iio.setFileName( save_as_file );
      iio.setFormat( "BMP" );
      iio.write();
   }
}

bool ImageViewer :: loadImage( const char *fileName )
{
   bool ok = FALSE;
   int w, h;

   if( filename )
   {
      QApplication :: setOverrideCursor( waitCursor );
      ok = image.load( filename , 0 );
      pm.convertFromImage( image , conversion_flags );

      if( ok )
      {
         setCaption( filename );
         w = pm.width();
         h = pm.height();
         h += menubar->heightForWidth(w) + status->height();
      }
      else
      {
         pm.resize( 0 , 0 );
         update();
      }

      setFixedWidth( w );
      setFixedHeight( h );
      status->setGeometry( 0, height() - status->height(),
            width(), status->height() );
      QApplication :: restoreOverrideCursor();
   }
   return ok;
```

```
}

void ImageViewer :: paintEvent( QPaintEvent *e )
{
   if( pm.size() != QSize( 0, 0 ) )
   {
      QPainter painter( this );
      painter.setClipRect( e->rect() );
      painter.drawPixmap( 0, menubar->heightForWidth(
            width() ), pm );
   }
}
```

Fişierul `main.cpp` conţine instanţierea unui obiect `QApplication` şi a unuia de tip `ImageViewer`.

```
#include "aplicatie.h"
#include <qapplication.h>
#include <qimage.h>

#ifdef QIMGIO
#include <qimageio.h>
#endif

int main( int argc, char **argv )
{
   QApplication :: setFont( QFont( "Helvetica" , 12 ) );
   QApplication a( argc, argv );

#ifdef QIMGIO
   qInitImageIO();
#endif

  ImageViewer *w = new ImageViewer( 0, "main window",
           QWidget :: WDestructiveClose ) ;
  w->show() ;
  QObject :: connect( qApp, SIGNAL( lastWindowClosed() ),
         qApp, SLOT(quit()) );
  return a.exec() ;
}
```

Lucrarea 3

Îmbunătăţirea imaginilor prin operaţii punctuale

BREVIAR TEORETIC

Termenul general de îmbunătăţire a imaginilor se referă la o clasă largă de operaţii, ce au ca scop mărirea detectabilităţii componentelor imaginii. Această detectabilitate depinde în primul rând de percepţia vizuală a unui observator uman şi deci reprezintă o apreciere subiectivă a imaginii.

Îmbunătăţirea calităţii unei imagini se face fără a presupune vreun model de degradare sau a lua în considerare vreo informaţie legată de imaginea originală. Paradoxal, dar şi o imagine originală (nedegradată) poate fi îmbunătăţită.

În general, îmbunătăţirea se referă la accentuarea unor caracteristici ale imaginii, cum ar fi muchiile, contururile sau contrastul. Procesul de îmbunătăţire nu măreşte cantitatea de informaţie conţinută de o imagine.

3.1 Operaţiile punctuale

Operaţiile punctuale sunt definite de o funcţie, care atribuie un nou nivel de gri pixelilor din imagine. Noua valoare a pixelului va depinde doar de vechea valoare a acestuia, de unde şi denumirea de "operaţie punctuală". Matematic se poate scrie:

$$g(k,l) = \phi(f(k,l)) \tag{3.1}$$

unde f este imaginea originală, iar g imaginea îmbunătăţită. $g(k,l)$ reprezintă noua valoare a pixelului (k,l), iar $f(k,l)$ vechea valoare. Operaţia punctuală este descrisă de funcţia ϕ.

O operaţie sau o transformare punctuală este reprezentată în Figura 3.1.

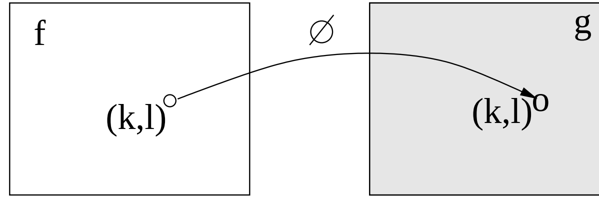


Figura 3.1: Reprezentarea unei operaţii punctuale.

3.2 Accentuarea contrastului

Această operaţie se foloseşte în special pentru îmbunătăţirea imaginilor cu contrast scăzut. Acesta poate apărea datorită unei slabe iluminări, a unei iluminări neuniforme sau din cauza unor neliniarităţi ale caracteristicilor unui senzor de captură a imaginii. O funcţie tipică de accentuare a contrastului are următoarea formă matematică:

$$\phi(x) = \begin{cases} \alpha x, & pentru \quad x \in [0, a) \\ \beta(x-a) + v_a, & pentru \quad x \in [a, b) \\ \gamma(x-b) + v_b, & pentru \quad x \in [b, L) \end{cases} \tag{3.2}$$

unde L reprezintă numărul de nivele pe care se face cuantizarea tonurilor de gri. Cazul cel mai frecvent este $L = 256$, pentru o cuantizare pe 8 biţi, x luând valori în intervalul [0,255]. Valorile α, β şi γ reprezintă pantele celor trei segmente de dreaptă.



Figura 3.2: Funcţia de accentuare de contrast.

După cum se observă din Figura 3.2, pantele α şi γ sunt pozitive şi subunitare, iar panta β este pozitivă şi supraunitară. Un segment cu pantă subunitară realizează o "apropiere" a nivelelor de gri, pe când un segment

cu pantă supraunitară va realiza o "depărtare" a nivelelor de gri, şi deci o accentuare a contrastului.

3.3 Întinderea maximă a contrastului

Aceasta este un caz particular al operaţiei de accentuare. Se foloseşte, în general, pentru reducerea zgomotului dintr-o imagine, atunci când se ştie că acesta ia valori cu precădere în intervalul $[a, b]$. Forma matematică este următoarea:

$$\phi(x) = \begin{cases} 0, & \textit{pentru} \quad x \in [0, a) \\ \beta(x-a), & \textit{pentru} \quad x \in [a, b] \\ L-1, & \textit{pentru} \quad x \in (b, L) \end{cases} \tag{3.3}$$

Nivelele de gri aflate în intervalul $[a, b]$ vor fi distanţate, ca urmare a pantei supraunitare, iar nivelele de gri ce se găsesc în afara acestui interval, vor fi înlocuite fie cu alb, fie cu negru, după caz.

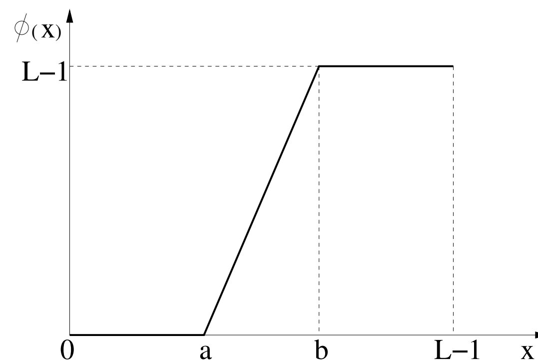


Figura 3.3: Funcţia de întindere maximă a contrastului.



(a) (b)

Figura 3.4: Întinderea maximă a contrastului: (a) imaginea originală şi (b) imaginea rezultată, pentru a=50, b=200.

3.4 Binarizarea imaginilor

Binarizarea sau prăguirea ("thresholding") imaginilor este un caz special al întinderii maxime a contrastului, pentru care $a = b$. Rezultatul operaţiei de binarizare este o imagine care conţine doar două nivele de gri: alb şi negru.

Pentru imagini în tonuri de gri, operaţia de binarizare se scrie matematic astfel:

$$\phi(x) = \begin{cases} 0, & pentru \quad x < T \\ L-1, & pentru \quad x \geq T \end{cases} \tag{3.4}$$

unde T este o valoare de prag, reprezentând o valoare întreagă din intervalul $[0, L)$.

Pentru imaginile color, binarizarea se poate face în următorul mod:

$$\phi(v) = \begin{cases} 0, & pentru \quad Y(v) < T \\ L-1, & pentru \quad Y(v) \geq T \end{cases} \tag{3.5}$$

unde v este un vector tridimensional ce reprezintă culoarea pixelului (de exemplu $v = (R, G, B)$), iar $Y(v)$ reprezintă luminanţa ($Y = 0.3R + 0.6G + 0.1B$).

În urma acestei transformări, contrastul este maximizat la nivelul întregii imagini.

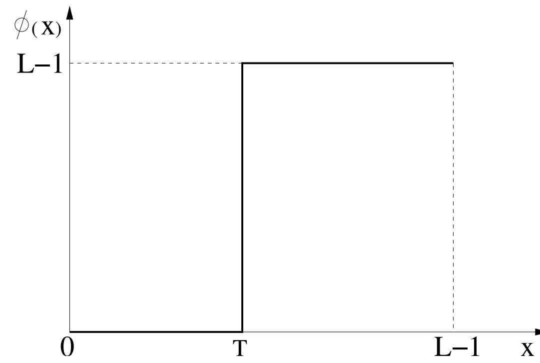


Figura 3.5: Funcţia de binarizare.



Figura 3.6: Binarizarea: (a) imaginea originală şi (b) imaginea binarizată cu T=127.

3.5 Negativarea imaginilor

Negativul unei imagini se obţine prin inversarea ordinii nivelelor de gri. Pentru imaginile în tonuri de gri, operaţia de negativare se face folosind funcţia:

$$\phi(x) = (L-1) - x \tag{3.6}$$

reprezentată în Figura 3.7, iar pentru imaginile color:

$$\phi(v) = ((L-1) - R, (L-1) - G, (L-1) - B) \tag{3.7}$$

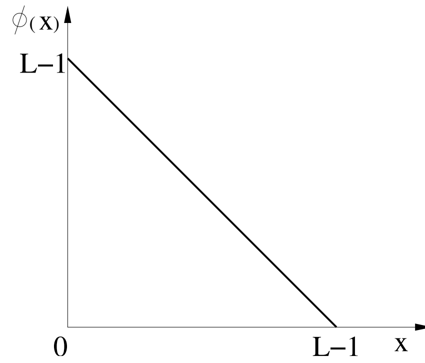


Figura 3.7: Funcţia de negativare.

Negativarea imaginilor este utilă pentru afişarea unor imagini medicale sau pentru realizarea de imagini negative pe suporturi fizice, de exemplu, de tip peliculă.



(a) (b)

Figura 3.8: Negativarea: (a) imaginea originală şi (b) negativul imaginii.

3.6 Decuparea imaginilor

Decuparea cu păstrarea fundalului este dată de formula:

$$\phi(x) = \begin{cases} L-1, & pentru \quad x \in [a,b] \\ x, & \hat{i}n\ rest. \end{cases} \tag{3.8}$$

iar decuparea fără păstrarea fundalului este dată de formula:

$$\phi(x) = \begin{cases} L-1, & pentru \quad x \in [a,b] \\ 0, & \hat{\imath}n\ rest. \end{cases} \tag{3.9}$$

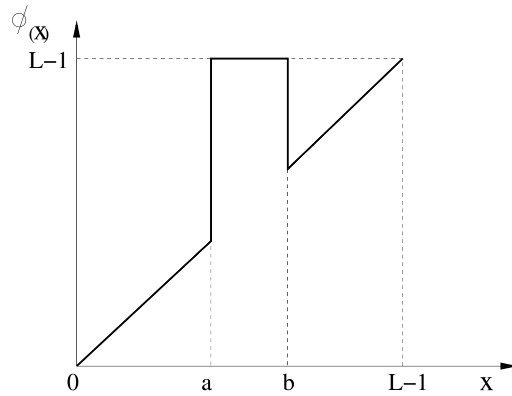


Figura 3.9: Funcţia de decupare cu păstrarea fundalului.

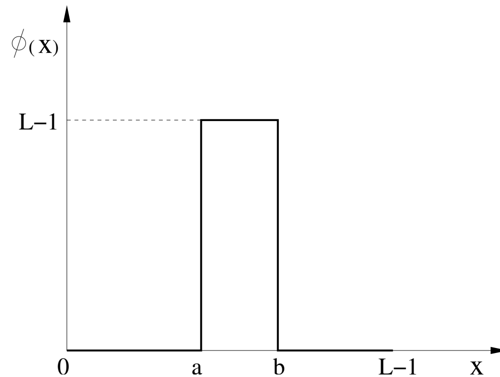


Figura 3.10: Funcţia de decupare fără păstrarea fundalului.

Aceste operaţii permit "decuparea" unor regiuni din imagine, caracterizate de anumite nivele de gri. Acest lucru este folosit atunci când diferite carcaterisitici ale imaginii sînt conţinute în nivelele de gri respective, cum ar fi de exemplu decuparea regiunilor de temperatură joasă reprezentate de nori din imaginile obţinute de un satelit meteo. În astfel de imagini, nivelele de gri ce corespund unor nori sînt direct proporţionale cu valorile de temperaturi joase.

DESFĂŞURAREA LUCRĂRII

Problema 1. Compilaţi sursele C++ ale lucrării. Rulaţi aplicaţia şi

observaţi rezultatul accentuării contrastului pentru o imagine în tonuri de gri (lena_AN.bmp). Funcţia care realizează accentuarea contrastului este prezentată în continuare, pentru următoarele valori: $a = 80$, $b = 170$, $V_a = 20$ şi $V_b = 235$.

Pantele care caracterizează funcţia de accentuare a contrastului vor fi următoarele: $\alpha = \frac{20}{80} = 0.4$, $\beta = \frac{235-20}{170-80} = \frac{215}{90}$ şi $\gamma = \frac{255-235}{255-170} = \frac{20}{85}$.

```
int ImageViewer :: f_accentuare( int nivel_gri )
{
   if( nivel_gri >= 0 && nivel_gri <= 80 )
      return ( int )( 0.4 * nivel_gri );

   if( nivel_gri > 80 && nivel_gri <= 170 )
      return ( int )( 215 / 90. * ( nivel_gri - 80 ) + 20 );

   if( nivel_gri > 170 && nivel_gri <= 255 )
      return ( int )( 20 / 85. * ( nivel_gri - 170 ) + 235 );

   return nivel_gri;
}

void ImageViewer :: accentueaza_contrastul( void )
{
   int w, h;
   int i, j;

   w = image.width();
   h = image.height();

   QImage imag_acc( w, h, 32, 0, QImage :: IgnoreEndian );

   for( i = 0; i < w; i++ )
      for( j = 0; j < h; j++ )
      {
         QRgb pixel = image.pixel( i, j );
         int gri_vechi = qRed( pixel );
         int gri_nou = f_accentuare( gri_vechi );
         imag_acc.setPixel( i, j,
            qRgb( gri_nou, gri_nou, gri_nou ) );
      }

   QImageIO iio;
   iio.setImage( imag_acc );
```

```
   iio.setFileName( "imag_acc.bmp" );
   iio.setFormat( "BMP" );
   iio.write();
}
```

Problema 2. Modificaţi valorile a, b, V_a şi V_b ale funcţiei de accentuare a contrastului, şi observaţi rezultatele accentuării contrastului.

Problema 3. Implementaţi operaţia de întindere maximă a contrastului.

Problema 4. Implementaţi operaţia de binarizare. Observaţi rezultatele acesteia pentru diferite valori ale pragului T.

Problema 5. Observaţi rezultatul operaţiei de negativare, pentru o imagine în tonuri de gri şi pentru o imagine color.

```
void ImageViewer :: negativeaza_imaginea( void )
{
   int w, h;
   int i, j;

   w = image.width();
   h = image.height();

   QImage imag_neg( w, h, 32, 0, QImage :: IgnoreEndian );

   for( i = 0; i < w; i++ )
      for( j = 0; j < h; j++ )
      {
         QRgb pixel = image.pixel( i, j );
         int r = qRed( pixel );
         int g = qGreen( pixel );
         int b = qBlue( pixel );
         imag_neg.setPixel( i, j, qRgb(255-r, 255-g, 255-b) );
      }

   QImageIO iio;
   iio.setImage( imag_neg );
   iio.setFileName( "imag_neg.bmp" );
   iio.setFormat( "BMP" );
   iio.write();
}
```

Problema 6. Implementaţi cele două operaţii de decupare.

Lucrarea 4

Îmbunătăţirea imaginilor prin egalizarea histogramei

BREVIAR TEORETIC

Tehnicile de îmbunătăţire a imaginilor bazate pe calculul histogramei modifică histograma astfel încât aceasta să aibă o anumită formă dorită.

4.1 Histograma unei imagini

Histograma unei imagini reprezintă frecvenţa relativă de apariţie a nivelelor de gri din imagine. Pentru o imagine f, de dimensiune $M \times N$ pixeli, histograma se defineşte astfel:

$$h(i) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \delta(i, f(m,n)) \;\;,\;\; i = 0, .., L-1 \tag{4.1}$$

unde funcţia δ este definită în următorul mod:

$$\delta(x,y) = \begin{cases} 1, & \text{\textit{dacă}} \;\; x = y, \\ 0, & \text{\textit{dacă}} \;\; x \neq y. \end{cases} \tag{4.2}$$

Din punct de vedere statistic, putem considera valoarea fiecărui pixel al imaginii ca o realizare particulară a unei variabile aleatoare asociată nivelelor de gri, caz în care histograma reprezintă funcţia de densitate de probabilitate a acestei variabile aleatoare. Fiind o funcţie de densitate de probabilitate, histograma oricărei imagini verifică condiţia de normare:

$$\sum_{i=0}^{L-1} h(i) = 1 \tag{4.3}$$

Practic, calculul histogramei presupune parcurgerea pixel cu pixel a imaginii şi contorizarea numărului de nivele de gri întâlnite.

4.2 Egalizarea histogramei

Egalizarea histogramei reprezintă o operaţie de accentuare a contrastului şi are ca scop obţinerea unei histograme uniforme.

Vom asocia unui pixel din imagine o variabilă aleatoare ξ. Astfel, valorile intensităţii luminoase ale pixelilor reprezintă realizări particulare ale variabilei aleatoare asociate. Vom considera că variabila aleatoare ξ are o densitate de probabilitate $w_\xi(x)$ şi o funcţie de repartiţie $F_\xi(x) = P\{\xi \leq x\}$.

Vom defini în continuare variabila η, care are funcţia de repartiţie:

$$F_\eta(x) = \int_0^x p_\xi(t)dt$$

şi care va fi uniform distribuită în intervalul $(0, 1)$.

Pentru cazul discret, presupunem că nivelele x de gri ale pixelilor au valori între 0 şi $L-1$ (unde L este de regulă 256), având asociate probabilităţile de apariţie $p_\xi(x_i)$, unde $x_i = 0, 1, ..., L-1$. Aceste probabilităţi pot fi estimate pe baza calculului histogramei, considerând imaginea dată, ca fiind o realizare particulară a procesului aleator descris de variabila aleatoare ξ, astfel:

$$p_\xi(x_i) = \frac{h(x_i)}{\sum\limits_{i=0}^{L-1} h(x_i)}$$

Noile nivele de gri, reprezentând valori discrete ale variabilei η din intervalul $[0, L-1]$ se vor calcula cu formulele:

$$h_c(x) = \sum_{x_i=0}^{x} p_\xi(x_i)$$

$$nivel_{nou} = int\left[\frac{h_c[nivel_{vechi}] - h_{cmin}}{1 - h_{cmin}}(L-1) + 0.5\right]$$

unde h_c reprezintă *histograma cumulativă* a imaginii, iar h_{cmin} este valoarea minimă a histogramei cumulative.

4.2.1 Algoritmul de egalizare a histogramei

Algoritmul de egalizare de histogramă, folosit în practică, poate fi descris în limbaj pseudocod astfel:

Pasul 1. Se calculează histograma imaginii:

```
pentru i = 1,..,L
        h[i] = 0
pentru i = 1,..,M
        pentru j = 1,..,N
```

nivel = imagine[i,j]
h[nivel] = h[nivel] + 1

unde L este numărul de nivele de gri (256), h este histograma imaginii, iar M şi N sunt dimensiunile imaginii.

Pasul 2. Se calculează histograma cumulativă a imaginii:

hc[1] = h[1]
pentru i = 2,..,L
hc[i] = hc[i-1] + h[i]

Pasul 3. Se calculează noile nivele de gri din imagine, sub forma unei transformări $y = T(x)$ dată de formula:

$$y = T(x) = \left[\frac{hc[x] - hc[1]}{NM - hc[1]} (L-1) + 0.5 \right]$$

astfel:

pentru i = 1,..,M
pentru j = 1,..,N
nivel_vechi = imagine[i,j]
nivel_nou = T(nivel_vechi)
imagine[i,j] = nivel_nou

4.3 Observaţii

- Deşi la prima vedere egalizarea de histogramă ar părea că este o operaţie punctuală, din cauza formulei de calcul a noilor valori de gri, ea este totuşi o operaţie integrală, datorită faptului că pentru fiecare pixel din imagine noua valoare se calculează pe baza calculului histogramei şi, deci, pe baza valorilor tuturor pixelilor din imagine.

- În Figura 4.1 puteţi observa cum imaginea a fost îmbunătăţită prin egalizarea histogramei.

- În Figura 4.2 puteţi observa cum s-a modificat forma histogramei imaginii originale, după egalizare.



Figura 4.1: Egalizarea histogramei: (a) imaginea originală şi (b) imaginea rezultată.

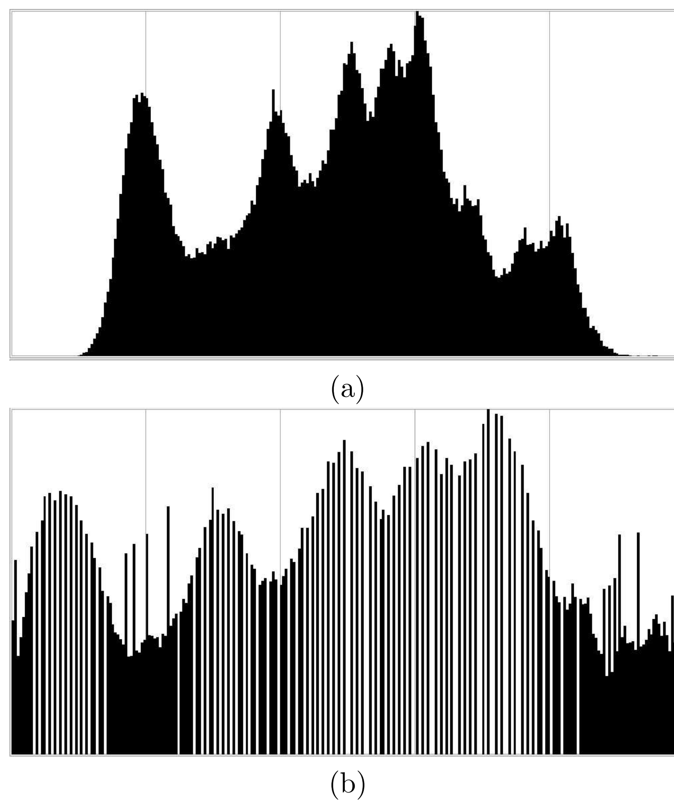


Figura 4.2: Histograma (a) originală şi (b) după egalizare.

DESFĂŞURAREA LUCRĂRII

O posibilă implementare a algoritmului de egalizare a histogramei este prezentat în continuare:

```
void ImageViewer :: egalizeaza_histograma( void )
{
   int i, j;
   int width, height;
   int h[ 256 ];

   for( i = 0; i < 256; i++ )
      h[ i ] = 0;

   width = image.width();
   height = image.height();

   //calcularea histogramei imaginii
   for( i = 0; i < width; i++ )
      for( j = 0; j < height; j++ )
      {
         QRgb pixel;
         pixel = image.pixel( i, j );

         int nivel_gri = qRed( pixel );
         h[ nivel_gri ]++ ;
      }

   //calcularea histogramei cumulative
   double hc[ 256 ];

   hc[ 0 ] = h[ 0 ];
   for( i = 1; i < 256; i++ )
   hc[ i ] = hc[ i - 1 ] + h[ i ];

   QImage imag_eq( width, height, 32, 0, QImage::IgnoreEndian );

   //egalizarea histogramei
   for( i = 0; i < width; i++ )
      for( j = 0; j < height; j++ )
      {
         QRgb pixel = image.pixel( i, j );

         int nivel = qRed( pixel );
```

```
         int nivel_nou = (int)( ( hc[nivel] - hc[0] ) * 255 /
             ( width*height - hc[0] ) );

         imag_eq.setPixel( i, j,
            qRgb( nivel_nou, nivel_nou, nivel_nou ) );
      }

   image = imag_eq;
   pm = image;
   update();
}
```

Problema 1. Observaţi forma histogramei pentru câteva imagini în tonuri de gri.

Problema 2. Observaţi efectele egalizării de histogramă pentru diferite imagini, inclusiv pentru o imagine subexpusă şi pentru una supraexpusă.

Problema 3. Modificaţi funcţia `histograma_imaginii` astfel încât aceasta să calculeze histograma cumulativă a imaginii. Observaţi forma unei histograme cumulative.

Problema 4. Justificaţi faptul că histograma cumulativă a unei imagini poate fi considerată estimatul unei funcţii de repartiţie.

Lucrarea 5

Transformări geometrice de bază

BREVIAR TEORETIC

Transformările geometrice sunt transformări care nu modifică valorile pixelilor din imagine, ci modifică doar aşezarea lor spaţială. Cu alte cuvinte, lasă nemodificată compoziţia imaginii, alterându-i însă structura. În urma aplicării unei transformări geometrice asupra unei imagini, histograma acesteia nu se modifică.

5.1 Translaţia

Translaţia se defineşte ca fiind operaţia de modificare *în linie dreaptă* a coordonatelor unui pixel din imagine de la o poziţie la alta. Un pixel de coordonate carteziene (x, y) va avea după translaţie coordonatele $(x^{'}, y^{'})$, astfel:

$$\begin{cases} x^{'} &= x + T_x \\ y^{'} &= y + T_y \end{cases} \tag{5.1}$$

unde perechea (T_x, T_y) reprezintă vectorul de translaţie (vezi Figura 5.1).

Translatarea unei imagini se realizează prin translatarea fiecărui pixel în parte. Valorile T_x şi T_y sunt numere întregi pozitive sau negative. În cazul în care noile coordonate ale unui pixel depăşesc dimensiunile imaginii, atunci el va fi poziţionat în partea opusă a imaginii, ca în Figura 5.2.

5.2 Rotaţia

Rotaţia se defineşte ca fiind operaţia de modificare *după o traiectorie circulară* a coordonatelor unui pixel din imagine (vezi Figura 5.3). Rotaţia

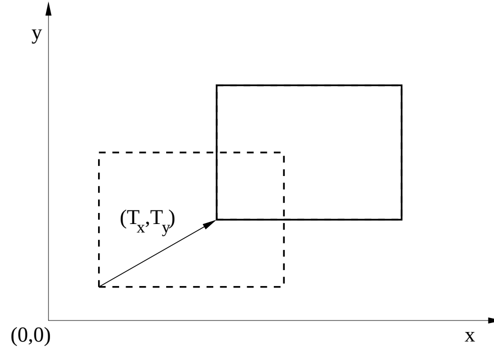


Figura 5.1: Translatarea unui obiect dreptunghiular.

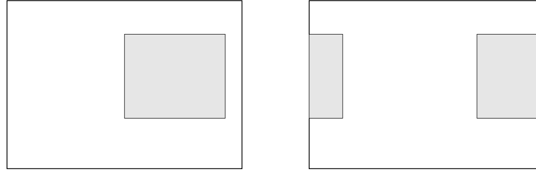


Figura 5.2: Translatarea spre dreapta a unui obiect dreptunghiular într-o imagine.

este specificată de unghiul ϕ. Poziţia unui pixel, exprimată în coordonate carteziene (x, y), se exprimă în coordonate polare (r, θ) astfel:

$$\begin{cases} x & = rcos\theta \\ y & = rsin\theta \end{cases} \tag{5.2}$$

Noile coordonate carteziene (x', y') ale pixelului rotit cu un unghi ϕ vor fi:

$$\begin{cases} x' & = rcos(\theta + \phi) = rcos\theta cos\phi - rsin\theta sin\phi = xcos\phi - ysin\phi \\ y' & = rsin(\theta + \phi) = rsin\theta cos\phi + rcos\theta sin\phi = xsin\phi + ycos\phi \end{cases} \tag{5.3}$$

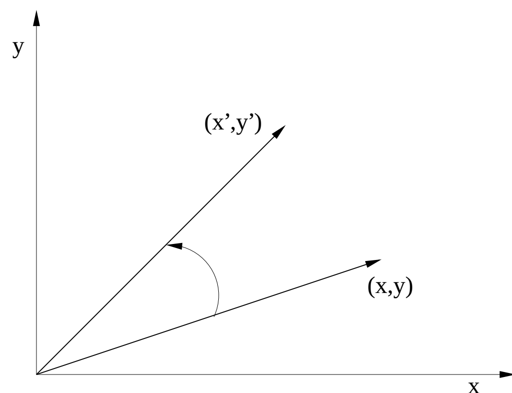


Figura 5.3: Rotaţia.

5.3 Oglindirea

Oglindirea este operaţia prin care se produce imaginea în oglindă a unui obiect, relativ la o axă de oglindire. În Figura 5.4 sunt ilustrate cele două feluri de oglindire: oglindirea "stânga-dreapta" şi cea "sus-jos".

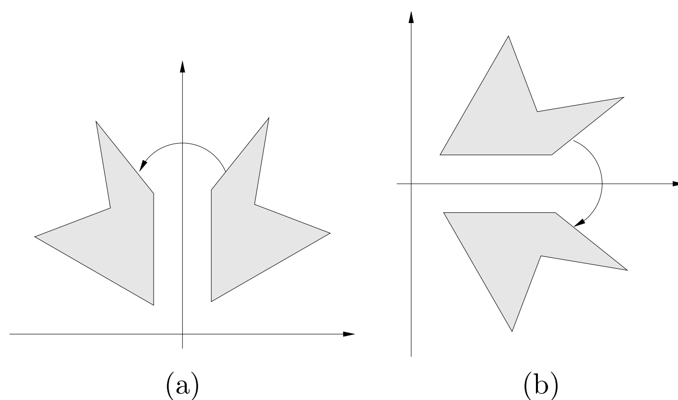


Figura 5.4: Oglindirea (a) "stânga-dreapta" (b) "sus-jos".

Observaţi în Figura 5.5 efectele oglindirii "stânga-dreapta" a imaginii Lena, faţă de o axă verticală ce trece prin centrul imaginii.



(a) (b)

Figura 5.5: Oglindirea "stânga-dreapta": (a) imaginea originală; (b) imaginea oglindită.

DESFĂŞURAREA LUCRĂRII

În continuare este prezentat codul C care implementează translaţia unei imagini pe orizontală, cu un deplasament de 100 de pixeli. Citiţi şi înţelegeţi codul.

```
void ImageViewer :: translateaza_imaginea( void )
{
   int w, h;
   int i, j;
   int iprim, tx = 100;

   w = image.width();
   h = image.height();

   QImage imag_tx( w, h, 32, 0, QImage :: IgnoreEndian );

   for( i = 0; i < w; i++ )
      for( j = 0; j < h; j++ )
      {
         QRgb pixel = image.pixel( i, j );

         iprim = i + tx;
         if( iprim >= w )
            iprim -= w;
```

```
            imag_tx.setPixel( iprim, j, pixel );
        }

    image = imag_tx;
    pm = image;
    update();
}
```

Problema 1. Implementaţi operaţia de translaţie, pentru următorii doi vectori de translaţie: (0,100) şi (100,100).

Problema 2. Implementaţi operaţia de oglindire “stânga-dreapta” faţă de axa verticală ce trece prin centrul imaginii.

Problema 3. Implementaţi operaţia de oglindire “sus-jos” faţă de axa orizontală ce trece prin centrul imaginii.

Lucrarea 6

Zgomotul în imagini

BREVIAR TEORETIC

Zgomotul este un semnal aleator, care afectează informaţia utilă conţinută într-o imagine. El poate apare de-alungul unui lanţ de transmisiune, sau prin codarea şi decodarea imaginii, şi reprezintă un element perturbator nedorit. De obicei se încearcă eliminarea lui prin diverse metode de filtrare.

Zgomotul se poate suprapune informaţiei utile în două moduri:

- **aditiv**. În acest caz, zgomotul se numeşte *zgomot aditiv* şi matematic se poate scrie:
$$g(x, y) = f(x, y) + n(x, y) \tag{6.1}$$
unde $f(x, y)$ este imaginea iniţială, neafectată de zgomot, $n(x, y)$ este zgomotul, iar $g(x, y)$ este imaginea afectată de zgomot.

- **multiplicativ**. În acest caz zgomotul se numeşte *zgomot multiplicativ*, iar fenomenul de suprapunere al acestuia peste informaţia utilă se scrie matematic astfel:
$$g(x, y) = f(x, y) \cdot n(x, y) \tag{6.2}$$
unde $f(x, y)$, $n(x, y)$ şi $g(x, y)$ au aceleaşi semnificaţii ca mai sus.

În continuare vom trata zgomotul ca fiind aditiv. Modelul de degradare a imaginii este reprezentat în figura 6.1. Zgomotul multiplicativ poate fi tratat la fel de simplu ca zgomotul aditiv, dacă logaritmăm relaţia (6.2):

$$log(g(x, y)) = log(f(x, y) \cdot n(x, y)) = log(f(x, y)) + log(n(x, y)) \tag{6.3}$$

Cantitativ, se poate aprecia măsura în care zgomotul a afectat informaţia utilă, calculând raportul semnal-zgomot[1]:

[1] SNR = (engl.) Signal Noise Ratio.

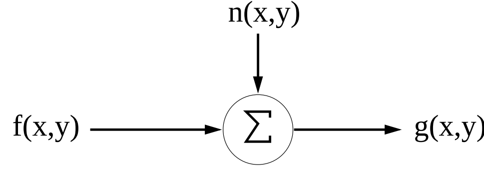


Figura 6.1: Modelul aditiv de degradare.

$$SNR = 10log\frac{\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} f^2(i,j)}{\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} n^2(i,j)}[dB] \tag{6.4}$$

$$SNR = 10log\frac{\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} f^2(i,j)}{\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} \left[f(i,j) - g(i,j)\right]^2}[dB] \tag{6.5}$$

Raportul semnal-zgomot reprezintă raportul dintre energia imaginii originale şi energia zgomotului suprapus acesteia. În continuare vom asocia valorilor pe care le ia zgomotul o variabilă aleatoare ξ, care, în funcţie de tipul zgomotului, va fi caracterizată de diferite funcţii de densitate de probabilitate.

6.1 Zgomotul cu distribuţie uniformă

Zgomotul cu distribuţie uniformă (vezi Figura 6.2) este caracterizat de o funcţie de densitate de probabilitate de forma:

$$w_\xi(x) = \begin{cases} A, & pentru\, x \in \left[-\frac{k}{2}; \frac{k}{2}\right], \\ 0, & în\, rest. \end{cases} \tag{6.6}$$

În Figura 6.3 puteţi observa efectele zgomotului cu distribuţie uniformă asupra imaginii “Lena”, pentru un raport semnal/zgomot de 5 dB.

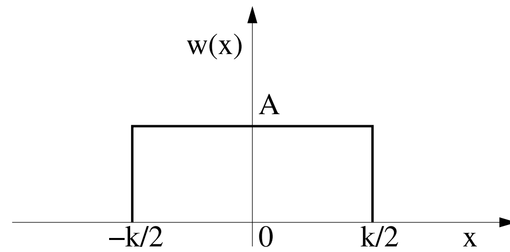


Figura 6.2: Funcţia de densitate de probabilitate pentru o distribuţie uniformă.



Figura 6.3: Zgomotul uniform: (a) imaginea originală; (b) imaginea afectată de zgomot uniform, SNR=5 dB.

6.2 Zgomotul cu distribuţie gaussiană

Zgomotul gaussian este caracterizat de o funcţie de densitate de probabilitate de forma (vezi Figura 6.4):

$$w_\xi(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{6.7}$$

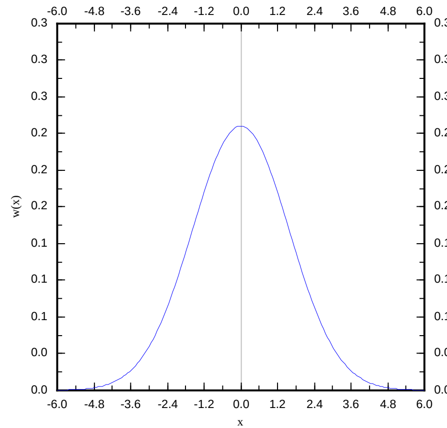


Figura 6.4: Funcţia de densitate de probabilitate pentru o distribuţie gaussiană.

Valoarea medie a unei variabilei aleatoare ξ, cu o distribuţie dată de funcţia $w_\xi(x)$ ca în relaţia (6.7), este μ, iar varianţa ei este σ^2. O distribuţie gaussiană se mai numeşte şi *normală*. O distribuţie normală de medie μ şi varianţă σ^2 se notează cu $N(\mu, \sigma^2)$.

În Figura 6.5 puteţi observa efectele zgomotului cu distribuţie gaussiană asupra imaginii "Lena", pentru un raport semnal/zgomot de 5 dB.



(a)



(b)

Figura 6.5: Zgomotul Gaussian: (a) imaginea originală; (b) imaginea afectată de zgomot gaussian, SNR=5 dB.

6.3 Zgomotul de tip "sare şi piper"

După cum îi spune numele, acest tip de zgomot va afecta valorile pixelilor în două moduri: "sare" - adică noua valoare a pixelului va fi 255 (pixelul va fi alb), sau "piper" - adică noua valoare a pixelului va fi 0 (pixelul va fi negru). Zgomotul de tip "sare şi piper" (vezi Figura 6.6) este perfect decorelat, deoarece între valorile 0 şi 255, şi între coordonatele pixelilor afectaţi de zgomot nu există corelaţie.



(a) (b)

Figura 6.6: Zgomotul "sare şi piper": (a) imaginea originală; (b) imaginea cu 10% pixeli afectaţi de zgomot.

6.4 Alte tipuri de zgomot

Zgomotele diferă între ele în funcţie de distribuţia care le caracterizează. Alte funcţii de distribuţie utilizate sunt:

- distribuţia Rayleigh: $w_\xi(x) = xe^{-\frac{x^2}{2}}$
- distribuţia Maxwell: $w_\xi(x) = x^2e^{-\frac{x^2}{2}}$
- distribuţia Beta: $w_\xi(x) = x^b(1-x)^c$
- distribuţia Gamma (Erlang): $w_\xi(x) = x^ne^{-x}$
- distribuţia Laplace: $w_\xi(x) = e^{-|x|}$
- distribuţia Cauchy: $w_\xi(x) = \frac{1}{1+x^2}$

Aceste funcţii sunt reprezentate în Figura 6.7.

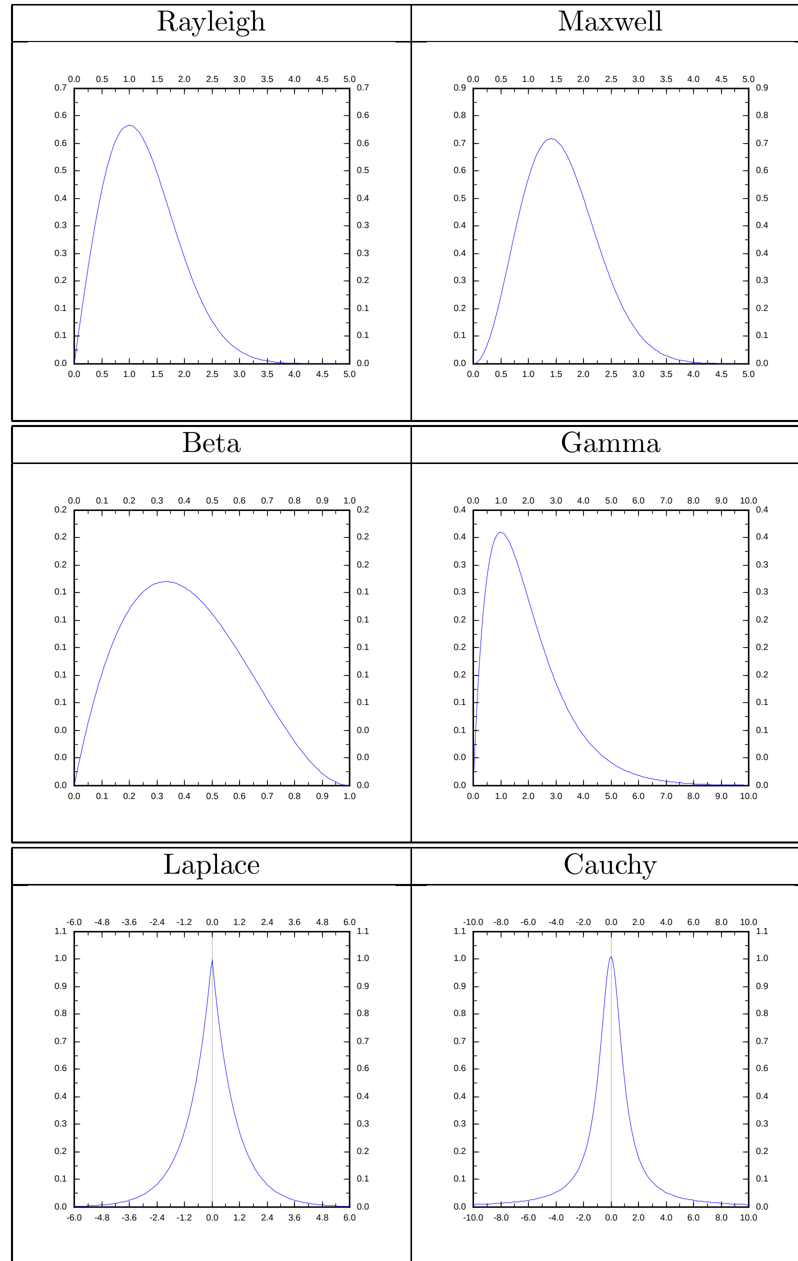


Figura 6.7: Diverse funcţii de distribuţie.

DESFĂŞURAREA LUCRĂRII

Problema 1. Suprapuneţi zgomot cu distribuţie uniformă peste o imagine în tonuri de gri (în meniul `Algoritmi`, funcţia `zgomot_uniform`).

```
void ImageViewer :: zgomot_uniform( void )
{
   int i, j;

   int w = image.width();
   int h = image.height();

   long int e_zgomot = 0; //energia zgomotului
   long int e_imagine = 0; //energia imaginii

   double SNR; //raportul semnal-zgomot

   //imaginea afectata de zgomot
   QImage image_aff( w, h, 32, 0, QImage::IgnoreEndian );

   int med = 0; //media zgomotului
   int dis = 200; //dispersia zgomotului

   for( i = 0; i < w; i++ )
      for( j = 0; j < h; j++ )
      {
         QRgb pixel = image.pixel( i, j );
         int nivel_gri = qRed( pixel );

         e_imagine += nivel_gri * nivel_gri;

         //srand( rand() );
         int zgomot = ( int )( med + sqrt( 3 )*
            (2. * ( rand()/( RAND_MAX + 1. )))*sqrt(dis) );
         e_zgomot += zgomot * zgomot;
         int val = nivel_gri + zgomot;

         if( val > 255 )
            val = 255;
         if( val < 0 )
            val = 0;

       image_aff.setPixel( i, j, qRgb( val, val, val ));
   }

   SNR = 10 * log( 1. * e_imagine / e_zgomot );

   image = image_aff;
   pm = image;
```

```
   update();

   QString mesaj;
   mesaj.sprintf( "SNR = %6.3lf dB", SNR );
   QMessageBox::about( this, "SNR", mesaj );
}
```

Problema 2. Modificaţi media şi dispersia zgomotului cu distribuţie uniformă (în fişierul `algoritmi.cpp`, funcţia `generează_zgomot_uniform`). Observaţi efectele.

Problema 3. Suprapuneţi zgomot cu distribuţie gaussiană peste o imagine în tonuri de gri (în meniul `Algoritmi`, funcţia `zgomot_gaussian`). Diferenţa între această funcţie şi cea care generează constă în modul în care este calculată valoarea zgomotului, şi anume:

```
int zgomot = ( int )( med + sqrt( 2 ) *
   sqrt( -1.* log( rand()/( RAND_MAX + 1. ))) *
   cos( 2 * 3.1415926 * ( rand()/( RAND_MAX+1 ))) *
   sqrt( dis ) );
```

Problema 4. Modificaţi media şi dispersia zgomotului cu distribuţie gaussiană (fişierul `algoritmi.cpp`, funcţia `generează_zgomot_gaussian`). Observaţi efectele.

Problema 5. Suprapuneţi zgomot de tip “sare şi piper” peste o imagine în tonuri de gri (în meniul `Algoritmi`, funcţia `zgomot_salt_and_pepper`). Codul acesteia este prezentat în continuare:

```
void ImageViewer :: salt_and_pepper( void )
{
   int i, j, k;
   int w = image.width();
   int h = image.height();

   double nr = 0.1; //procentul de pixeli afectati de zgomot

   srand( rand() );

   k = 0;
   while( k < ( int )( w * h * nr ) )
   {
      i = ( int )( 1. * w * rand() / ( RAND_MAX + 1. ) );
      j = ( int )( 1. * h * rand() / ( RAND_MAX + 1. ) );

      QRgb sare_piper;
```

```
      if( ( 100. * rand() / ( RAND_MAX + 1. ) ) >= 50 )
         sare_piper = qRgb( 255, 255, 255 );
      else
         sare_piper = qRgb( 0, 0, 0 );

      if( (i >= 0) && (i < w) && (j >= 0) && (j < h) )
         image.setPixel( i, j, sare_piper );

      k++;
   }

   pm = image;
   update();
}
```

Problema 6. Modificaţi procentul de pixeli afectaţi de zgomot de tip "sare şi piper" (fişierul `algoritmi.cpp`, funcţia `zgomot_salt_and_pepper`). Observaţi efectele.

Lucrarea 7

Filtrarea imaginilor

BREVIAR TEORETIC

Filtrarea imaginilor se înscrie în clasa operaţiilor de îmbunătăţire, principalul scop al acesteia fiind eliminarea zgomotului suprapus unei imagini.

Filtrarea reprezintă o operaţie de vecinătate. Prin aceasta se înţelege că la calculul noii valori a unui pixel vor contribui şi valorile pixelilor vecini, nu doar vechea lui valoare, cum se întâmpla la operaţiile punctuale. Vecinii unui pixel reprezintă o mulţime de pixeli, aflaţi în apropierea acestuia, care alcătuiesc o *vecinătate*. Această vecinătate poate avea diverse forme şi dimensiuni, însă cele mai utilizate în practică sînt vecinătăţile de formă pătrată, de dimensiuni impare.

7.1 Filtrarea liniară a imaginilor

După cum îi spune numele, acest tip de filtrare respectă principiul liniarităţii (sau al superpoziţiei).
Principiul liniarităţii: Fiind date două imagini $f_1(x, y)$ şi $f_2(x, y)$, şi două numere reale α şi β, se numeşte **operator liniar**, un operator O care are următoarea proprietate:

$$O\left[\alpha \cdot f_1(x,y) + \beta \cdot f_2(x,y)\right] = \alpha \cdot O[f_1(x,y)] + \beta \cdot O[f_2(x,y)] \tag{7.1}$$

Operaţia de filtrare liniară calculează noua valoare a unui pixel al imaginii, din poziţia (m,n), ca o combinaţie liniară a unui număr de valori din imaginea originală, astfel:

$$g(m,n) = \sum_{(k,l)} \sum_{\in W} w_{kl} \cdot f(m-k, n-l) \tag{7.2}$$

unde $f(x, y)$ este imaginea originală (afectată sau nu, de zgomot), $g(x, y)$ este imaginea filtrată, W este o structură de puncte care defineşte vecinătatea

pixelului (m, n), w_{kl} sînt nişte valori constante care reprezintă **coeficienţii filtrului**.

Filtrul este definit de vecinătatea W şi de coeficienţii w_{kl}. Un filtru poate fi specificat de o matrice V, care poartă numele de **mască de convoluţie** sau **mască de filtrare**, care este caracterizată de formă, valorile coeficienţilor şi de origine. În Figura 7.1 este prezentată o mască de filtrare de formă pătrată, de dimensiune 3x3, având originea în centru.

$w_{-1,-1}$	$w_{-1,0}$	$w_{-1,1}$
$w_{0,-1}$	$\boldsymbol{w_{0,0}}$	$w_{0,1}$
$w_{1,-1}$	$w_{1,0}$	$w_{1,1}$

Figura 7.1: Mască de filtrare pătrată de dimensiune 3x3.

Nu este obligatoriu ca forma măştii de filtrare să fie pătrată, de dimensiune impară sau să aibă originea în centrul măştii.

Operaţia de filtrare liniară poate fi descrisă astfel: se suprapune masca de filtrare peste fiecare pixel al imaginii originale, astfel încât originea măştii să coincidă cu pixelul considerat, apoi se calculează toate produsele între coeficienţii măştii şi valorile pixelilor peste care se suprapun aceşti coeficienţi, iar suma acestor produse reprezintă noua valoare a pixelului considerat. Această tehnică poartă numele de **tehnica ferestrei glisante**. Operaţia descrisă reprezintă de fapt o convoluţie bidimensională.

7.1.1 Filtrele de netezire

Filtrele de netezire sunt echivalentele bidimensionale ale filtrelor trece-jos (FTJ), şi la fel ca acestea, sînt folosite în general pentru eliminarea zgomotului, care se presupune că este de bandă largă.

Informaţia conţinută într-o imagine, în general, se regăseşte în componentele de joasă frecvenţă, şi deci este justificată o filtrare trece-jos pentru reducerea puterii zgomotului.

Zgomotul din imagine se presupune că este aditiv şi pur aleator, adică se consideră următoarele ipoteze simplificatoare:

- $g(i, j) = f(i, j) + n(i, j)$ (zgomotul n este aditiv),
- n este un semnal staţionar (comportamentul său statistic nu depinde de coordonatele i şi j ale pixelului),

- $\overline{n} = 0$ (media zgomotului este zero),
- dacă zgomotul are dispersia σ_n, atunci:

$$\overline{n(i_1, j_1) \cdot n(i_2, j_2)} = \begin{cases} \sigma_n^2 & pentru \quad i_1 = i_2 \;\&\; j_1 = j_2 \\ 0 & \hat{i}n\ rest \end{cases}$$

(zgomotul este complet decorelat).

Condiţia de normare a coeficienţilor filtrelor de netezire

Pentru un filtru trece-jos se impune următoarea condiţie: componenta continuă a imaginii să nu fie alterată de filtru. Cu alte cuvinte, filtrul să conserve luminozitatea medie a imaginii.

Pentru aceasta, considerăm o imagine având un singur nivel de gri, constant, notat cu μ, adică: $f(i, j) = \mu$ pentru oricare i şi j. Pentru ca filtrul să conserve luminozitatea medie, adică valoarea medie μ, impunem $g(i, j) = \mu$ pentru $\vee i, j$. Rezultă:

$$\mu = \sum_{(k,l)} \sum_{\in W} w_{kl} \cdot \mu \tag{7.3}$$

$$\sum_{(k,l)} \sum_{\in W} w_{kl} = 1 \tag{7.4}$$

unde $w_{kl} \geq 0$.

Relaţia (7.4) poartă numele de condiţie de normare pentru filtre de netezire (sau trece-jos).

Filtrul de mediere

Filtrul de mediere este cel mai simplu filtru de netezire. Caracteristic unui filtru de mediere este faptul că toţi coeficienţii măştii de filtare sînt egali. Dacă ţinem cont şi de condiţia de normare, atunci coeficienţii unui filtru de mediere, care are o mască de filtrare de dimensiune $N \times N$, au valoarea $\frac{1}{N^2}$. În Figurile 7.2, 7.3 şi 7.4 sunt prezentate măştile de filtrare de mediere, pentru N = 3, 5 şi respectiv 7.

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\mathbf{\frac{1}{9}}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

Figura 7.2: Mască de filtrare pătrată de dimensiune 3×3.

$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$
$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$
$\frac{1}{25}$	$\frac{1}{25}$	$\mathbf{\frac{1}{25}}$	$\frac{1}{25}$	$\frac{1}{25}$
$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$
$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$

Figura 7.3: Mască de filtrare pătrată de dimensiune 5×5.

$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$
$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$
$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$
$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$	$\mathbf{\frac{1}{49}}$	$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$
$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$
$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$
$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$	$\frac{1}{49}$

Figura 7.4: Mască de filtrare pătrată de dimensiune 7×7.



(a)



(b)

Figura 7.5: Filtrarea de mediere: (a) imaginea originală; (b) imaginea filtrată cu o mască 7x7.

Filtrul de mediere nu este folosit în practică, deoarece, pe lângă zgomot, are de suferit şi semnalul util (vezi Figura 7.5). Filtrarea de mediere este deranjantă pentru imagine în acele zone în care imaginea conţine frecvenţe înalte (variaţii bruşte), pentru că duce la apariţia fenomenului de înceţoşare[1].

[1](engl.) blurring.

Din punctul de vedere al zgomotului, pentru filtrare este utilă o mască de filtrare de dimensiune cât mai mare. Din punctul de vedere al semnalului util, al imaginii, este util ca masca să fie cât mai mică. În practică se realizează un compromis între cele două aspecte.

Alte măşti de filtrare

Alte măşti de filtrare, pentru filterele de mediere, sunt prezentate în continuare, deşi toate au acelaşi randament nesatisfăcător.

$$\begin{pmatrix} \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \\ \frac{2}{16} & \frac{4}{16} & \frac{2}{16} \\ \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \end{pmatrix} \begin{pmatrix} \frac{1}{16} & \frac{1}{16} & \frac{1}{16} \\ \frac{1}{16} & \frac{1}{2} & \frac{1}{16} \\ \frac{1}{16} & \frac{1}{16} & \frac{1}{16} \end{pmatrix} \begin{pmatrix} 0 & \frac{1}{5} & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ 0 & \frac{1}{5} & 0 \end{pmatrix} \begin{pmatrix} 0 & \frac{1}{8} & 0 \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ 0 & \frac{1}{8} & 0 \end{pmatrix}$$

7.1.2 Filtrele trece-sus

Filtrele trece-sus urmăresc eliminarea componentelor de frecvenţă joasă din imagine. Sunt folosite în general pentru detectarea frontierelor sau contururilor din imagine, acolo unde au loc treceri sau variaţii bruşte ale luminanţei.

Condiţia de normare a coeficienţilor filtrelor trece-sus

Condiţia de normare a coeficienţilor pentru un filtru trece-jos (relaţia (7.6)) se determină impunând condiţia ca filtrul să rejecteze complet (sau să atenueze complet) componenta continuă a imaginii.

Pentru aceasta vom considera, la fel, o imagine $f(i,j) = \mu$ pentru $\vee i, j$. La ieşirea filtrului trece-sus vom avea $g(i,j) = 0$ pentru $\vee i, j$.

$$0 = \sum_{(k,l)} \sum_{\in W} w_{kl} \cdot \mu \tag{7.5}$$

$$\sum_{(k,l)} \sum_{\in W} w_{kl} = 0 \tag{7.6}$$

Filtrul de accentuare

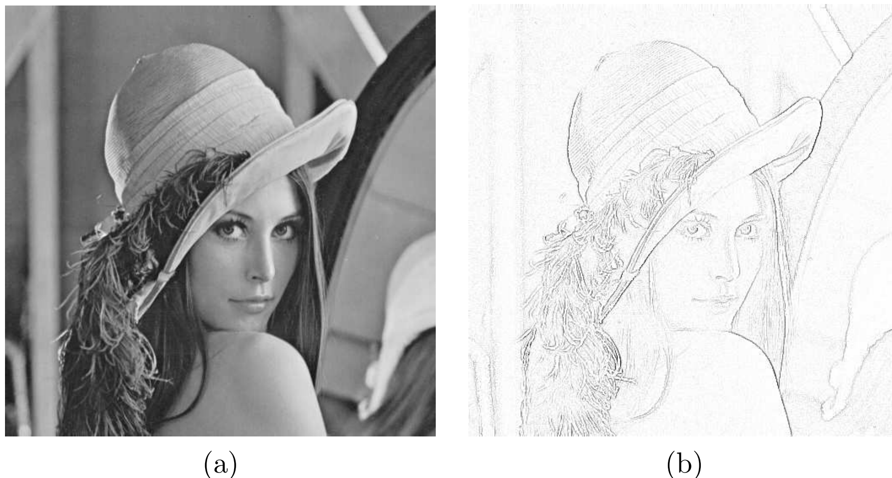
Filtrul de accentuare nu este un filtru trece-sus, ci foloseşte filtrarea trece-sus pentru a realiza accentuarea. Prin accentuare se înţelege contrastarea unei imagini şi are ca scop îmbunătăţirea perceperii vizuale a contururilor obiectelor. Cu alte cuvinte, îmbunătăţirea detectabilităţii componentelor scenei de-a lungul frontierelor acestora. Acest lucru se realizează, în principiu, prin modificarea valorilor pixelilor situaţi de o parte şi de alta a unei frontiere comune.

Sistemul uman are tendinţa de a adânci profilul zonelor de tranziţie dintre regiuni uniforme. Studiul fiziologiei sistemului vizual a demonstrat că aceasta se realizează prin prelucrări de tip derivativ ce apar în diferitele etape pe care le parcurge informaţia vizuală. Efectul global poate fi modelat matematic prin scăderea din semnalul original a unei derivate secunde ponderate.

În continuare sunt prezentate câteva măşti de implementare a unei derivate secunde de tip Laplace:

$$\begin{pmatrix} 0 & -\frac{1}{4} & 0 \\ -\frac{1}{4} & 1 & -\frac{1}{4} \\ 0 & -\frac{1}{4} & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{4} & -\frac{1}{2} & \frac{1}{4} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ \frac{1}{4} & -\frac{1}{2} & \frac{1}{4} \end{pmatrix} \begin{pmatrix} -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} \\ -\frac{1}{8} & 1 & -\frac{1}{8} \\ -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} \end{pmatrix}$$

În Figura 7.6 puteţi observa efectele filtrării Laplace.



(a) (b)

Figura 7.6: Filtrarea Laplace: (a) imaginea originală; (b) imaginea rezultată (negativată).

Filtrul de accentuare se implementează după schema prezentată în Figura 7.7.

7.2 Filtrarea neliniară a imaginilor

Filtrele neliniare nu respectă principiul liniarităţii sau al superpoziţiei enunţat la începutul lucrării. Acestea au apărut din necesitatea de a depăşi limitările filtrelor liniare, în ceea ce priveşte zgomotele care nu au o distribuţie normală sau nu sunt aditive.

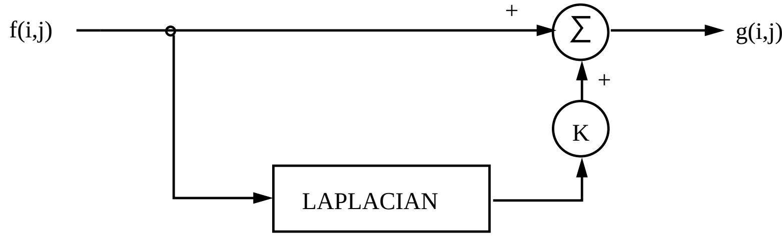


Figura 7.7: Filtrul de accentuare.

7.2.1 Filtrele de ordine

Filtrele de ordine sunt operatori locali, definiţi la rândul lor de o fereastră, care selectează din imagine un număr de pixeli vecini pixelului curent, într-un mod identic cu tehnica ferestrei glisante. Valorile pixelilor selectaţi se ordonează crescător.

Să presupunem că fereastra conţine n pixeli, iar valorile lor formează următoarea mulţime:

$$\{x_1, x_2, ..., x_n\} \tag{7.7}$$

După ce aceste valori au fost ordonate crescător, vom avea:

$$\{x_{(1)}, x_{(2)}, ..., x_{(n)}\} \tag{7.8}$$

pentru care sunt îndeplinite următoarele condiţii:

$$x_{(1)} \leq x_{(2)} \leq ... \leq x_{(n)} \tag{7.9}$$

Ieşirea filtrului de ordine de ordin k, pentru $k \in [1; n]$ întreg, este statistica de ordinul k, cu alte cuvinte, elementul de pe poziţia k din şirul ordonat:

$$rank_k\{x_1, x_2, ..., x_n\} = x_{(k)} \tag{7.10}$$

Filtrul median

Filtrul median ete un filtru de ordine a cărui ieşire este statistica de ordin central, adică elementul ce se află pe poziţia de mijloc a şirului ordonat de valori selectate de fereastra de filtrare:

$$median\{x_1, x_2, ..., x_n\} = \begin{cases} x_{(\frac{n+1}{2})} & dacă\ n\ este\ impar, \\ \frac{1}{2}x_{(\frac{n}{2})} + \frac{1}{2}x_{(\frac{n}{2}+1)} & dacă\ n\ este\ par. \end{cases} \tag{7.11}$$

Filtrul median este potrivit pentru eliminarea zgomotului de tip “sare şi piper”. După ordonarea valorilor pixelilor, valorile zgomotului (adică 0 sau 255) se vor situa pe primele, respectiv ultimele poziţii în mulţime, şi deci, la ieşirea filtrului, vom avea o valoare diferită de valorile zgomotului. Totuşi există şi situaţii în care, după filtrare, mai există pixeli afectaţi de zgomot. În acest caz, spunem că filtrul a fost “străpuns” de zgomot. Acest lucru este posibil atunci când mai mult de jumătate din pixelii selectatţi de fereastra de filtrare, sunt afectaţi în acelaşi mod (sare sau piper, 255 sau 0) de zgomot.



(a)



(b)

Figura 7.8: Filtrul median: (a) imaginea originală afectată de zgomot “sare şi piper”; (b) imaginea filtrată.

Filtrul de minim

Filtrul de minim este un filtru de ordine a cărui ieşire este statistica de ordinul 1, adică valoarea $x_{(1)}$, care este cea mai mică valoare din mulţimea de valori ale pixelilor selectaţi de către fereastra de filtrare.

Filtrul de maxim

Filtrul de maxim este un filtru de ordine a cărui ieşire este statistica de ordinul n, adică valoarea $x_{(n)}$, care este cea mai mare valoare din mulţimea de valori luate în considerate.

DESFĂŞURAREA LUCRĂRII

Problema 1. Observaţi efectele filtrului de mediere pentru o imagine afectată de zgomot gaussian. Codul C al funcţiei care implementează filtrul de

mediere este prezentat în continuare:

```
void ImageViewer :: filtru_mediere( void )
{
   int i, j;
   int k, l;
   int w, h;

   double v[ 3 ][ 3 ];

   //coeficientii mastii de filtrare
   v[0][0] = 1./9;  v[0][1] = 1./9;  v[0][2] = 1./9;
   v[1][0] = 1./9;  v[1][1] = 1./9;  v[1][2] = 1./9;
   v[2][0] = 1./9;  v[2][1] = 1./9;  v[2][2] = 1./9;

   w = image.width();
   h = image.height();

   QImage image_fil( w, h, 32, 0, QImage::IgnoreEndian );

   for( i = 1; i < w - 1; i++ )
      for( j = 1; j < h - 1; j++ )
      {
         //suma ponderata
         double sum = 0;

         for( k = -1; k < 2; k++ )
            for( l = -1; l < 2; l++ )
               sum += v[ k + 1 ][ l + 1 ] *
         qRed( image.pixel( i + k, j + l ));

         image_fil.setPixel( i, j,
            qRgb( (int)sum, (int)sum, (int)sum ));
      }

   image = image_fil;
   pm = image;
   update();
}
```

Problema 2. Observaţi efectele filtrului de mediere pentru o imagine afectată de zgomot de tip “sare şi piper”.

Problema 3. Observaţi efectul de “blurring” al filtrului de mediere pentru o imagine neafectată de zgomot.

Problema 4. Implementaţi un filtru de mediere cu o mască de filtrare de formă pătrată de dimensiune 5x5.

Problema 5. Observaţi efectul filtrului de accentuare (pentru o imagine neafactată de zgomot). Codul C al filtrului este următorul:

```
void ImageViewer :: filtru_accentuare( void )
{
   int i, j;
   int k, l;
   int w, h;

   double sum;
   double v[ 3 ][ 3 ];

   //coeficientii mastii
   v[0][0] = 0;     v[0][1] = -1./4; v[0][2] = 0;
   v[1][0] = -1./4; v[1][1] = 1;     v[1][2] = -1./4;
   v[2][0] = 0;     v[2][1] = -1./4; v[2][2] = 0;

   w = image.width();
   h = image.height();

   QImage image_fil( w, h, 32, 0, QImage::IgnoreEndian );

   for( i = 1; i < w - 1; i++ )
      for( j = 1; j < h - 1; j++ )
      {
         sum = 0;

         for( k = -1; k < 2; k++ )
            for( l = -1; l < 2; l++ )
               sum += 1. * v[ k + 1 ][ l + 1 ] *
                  qRed( image.pixel( i + k, j + l ));

         int niv = qRed( image.pixel( i, j ));
         niv = (int)( niv + 0.6 * sum );
         image_fil.setPixel( i, j, qRgb( niv, niv, niv ));
      }

   image = image_fil;
   pm = image;
   update();
}
```

Problema 6. Observaţi efectele filtrului median pentru o imagine afectată de zgomot de tip "sare şi piper". Citiţi şi înţelegeţi implementarea în C:

```
void ImageViewer :: filtru_median( void )
{
   int i, j;
   int w, h;
   int k, aux;
   int m, n;
   int med;
   int sir[ 9 ];

   w = image.width();
   h = image.height();

   QImage image_fil( w, h, 32, 0, QImage::IgnoreEndian );

   for( i = 1; i < w-1; i++ )
      for( j = 1; j < h-1; j++ )
      {
          //formarea unui sir din elementele vecinatatii 3x3
          k = 0;
          for( m = -1; m < 2; m++ )
             for( n = -1; n < 2; n++ )
             {
                 sir[k] = qRed( image.pixel( i+m, j+n ) );
                 k++;
             }

          //ordonarea crescatoare a valorilor pixelilor
          //metoda BUBBLE SORT
          k = 0;
          while( k == 0 )
          {
             k = 1;
             for( m = 0; m < 8; m++ )
                if( sir[ m ] > sir[ m + 1 ] )
                {
               aux = sir[ m ];
                    sir[ m ] = sir[ m + 1 ];
                    sir[ m + 1 ] = aux;
                    k = 0;
                }
```

```
            }

            //elementul median
            med = sir[ 4 ];

            //noua valoare a pixelului
            image_fil.setPixel( i, j, qRgb( med, med, med ) );
    }

    image = image_fil;
    pm = image;
    update();
}
```

Problema 7. Implementaţi filtrul de minim. Observaţi efectele lui asupra unei imagini neafectate de zgomot.

Problema 8. Implementaţi filtrul de maxim. Observaţi efectele lui asupra unei imagini neafectate de zgomot.

Lucrarea 8

Transformări unitare

BREVIAR TEORETIC

Transformările reprezintă o categorie de prelucrări ce include operaţii de tip integral, la calculul noii valori a unui pixel al imaginii transformate contribuind valorile tuturor pixelilor din imaginea originală.

Termenul de transformare se referă la o clasă de matrici unitare folosite pentru a reprezenta imagini. O matrice A de dimensiune $N \times N$ este unitară dacă inversa ei este matricea A^{*T}:

$$A \cdot A^{-1} = A \cdot A^{*T} = A^{*T} \cdot A = I_N \tag{8.1}$$

unde * reprezintă operaţia de complementare în mulţimea numerelor complexe, ^{T} reprezintă operaţia de transpunere a unei matrici, iar I_N este matricea identitate de dimensiune $N \times N$:

$$I_N = \begin{pmatrix} 1 & 0 & 0 & ... & 0 \\ 0 & 1 & 0 & ... & 0 \\ 0 & 0 & 1 & ... & 0 \\ . & . & . & ... & . \\ . & . & . & ... & . \\ 0 & 0 & 0 & ... & 1 \end{pmatrix} \tag{8.2}$$

Pentru un vector uni-dimensional $\underline{u}$ de dimensiune N, de forma:

$$\underline{u} = \begin{bmatrix} u(0) \\ u(1) \\ . \\ . \\ . \\ u(N-1) \end{bmatrix} = [u(0), u(1), ..., u(N-1)]^T \tag{8.3}$$

se numeşte **transformare unitară directă** relaţia:

$$v(k) = \sum_{n=0}^{N-1} a(k,n) \cdot u(n), \quad 0 \leq k \leq N-1 \tag{8.4}$$

unde $v(k)$ reprezintă elementele vectorului transformat $\underline{v}$, iar $a(k,n)$ sînt elementele matricii A. Matricial această relaţie se poate scrie astfel:

$$\underline{v} = A \cdot \underline{u} \tag{8.5}$$

Transormarea unitară inversă este dată de relaţia:

$$u(n) = \sum_{k=0}^{N-1} v(k) \cdot a^*(k,n), \quad 0 \leq n \leq N-1 \tag{8.6}$$

care se scrie matricial astfel:

$$\underline{u} = A^{*T} \cdot \underline{v} \tag{8.7}$$

Valorile vectorului $\underline{v}$ sunt o reprezentare a vectorului iniţial $\underline{u}$, într-un alt spaţiu. O astfel de reprezentare este utilă în filtrare, compresie, extragere de trăsături sau alte tipuri de analiză a imaginilor.

8.1 Transformări unitare bidimensionale

Pentru o imagine $u(m,n)$, de dimensiune $N \times N$, transformarea directă are următoarea formă:

$$v(k,l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} u(m,n) \cdot a_{k,l}(m,n), \quad 0 \leq k,l \leq N-1 \tag{8.8}$$

iar transformarea inversă:

$$u(m,n) = \sum_{k=0}^{N-1} \sum_{k=0}^{N-1} v(k,l) \cdot a^*_{k,l}(m,n), \quad 0 \leq m,n \leq N-1 \tag{8.9}$$

unde coeficienţii $\{a_{k,l}(m,n)\}$ poartă numele de **transformare unitară bidimensională**, şi reprezintă un set de matrici de bază ortonormale, iar $v(k,l)$ reprezintă transformata imaginii $u(m,n)$.

Aceste matrici de bază respectă condiţia de **ortonormalitate**:

$$\sum_{m=0}^{N-1} \sum_{n=0}^{N-1} a_{k,l}(m,n) \cdot a^*_{k',l'}(m,n) = \delta(k-k', l-l') = \begin{cases} 1, & k = k' \;\&\; l = l', \\ 0, & \textit{în rest.} \end{cases} \tag{8.10}$$

pentru $\vee k, l$.

O transformare ca cea dată de relaţia (8.8) este caracterizată de N^4 coeficienţi $a_{k,l}(m,n)$. Pentru calculul unui singur element $v(k,l)$ (k şi l fixaţi) este nevoie de un număr de N^2 înmulţiri. Prin urmare complexitatea întregului calcul este $O(N^4)$. Complexitatea poate fi redusă la $O(N^3)$ dacă transformarea este separabilă.

O transformare este **separabilă**, dacă elementele $a_{k,l}(m,n)$ ce o definesc, pot fi scrise ca produs de alte două elemente, grupate după perechi de indici, astfel:

$$a_{k,l}(m,n) = a_k(m) \cdot b_l(n) = a(k,m) \cdot b(l,n) \tag{8.11}$$

unde, matricile $A = \{a(k,m)\}$ şi $B = \{b(l,n)\}$ trebuie să fie la rândul lor unitare, adică:

$$A \cdot A^{*T} = A^{*T} \cdot A = I_N \tag{8.12}$$

$$B \cdot B^{*T} = B^{*T} \cdot B = I_N \tag{8.13}$$

Dacă transformarea este separabilă, atunci relaţiile (8.8) şi (8.9) devin:

$$v(k,l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} a(k,m) \cdot u(m,n) \cdot b(l,n) \tag{8.14}$$

$$u(m,n) = \sum_{k=0}^{N-1} \sum_{k=0}^{N-1} a^*(k,m) \cdot v(k,l) \cdot b^*(l,n) \tag{8.15}$$

care pot fi scrise matricial astfel:

$$V = A \cdot U \cdot B^T \tag{8.16}$$

$$U = A^{*T} \cdot V \cdot B^* \tag{8.17}$$

unde $U = \{u(m,n)\}$ reprezintă imaginea originală, iar $V = \{v(k,l)\}$ reprezintă imaginea transformată.

În practică se folosesc numai transformări separabile, pentru care, în plus, se alege $B = A$. În acest caz, vom avea o singură matrice A, unitară, care defineşte transfomarea, iar relaţiile (8.14) şi (8.15) devin:

$$v(k,l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} a(k,m) \cdot u(m,n) \cdot a(l,n) \tag{8.18}$$

$$u(m,n) = \sum_{k=0}^{N-1} \sum_{k=0}^{N-1} a^*(k,m) \cdot v(k,l) \cdot a^*(l,n) \tag{8.19}$$

Matricial, relaţiile (8.16) şi (8.17) se scriu după cum urmează:

$$V = A \cdot U \cdot A^T \tag{8.20}$$

$$U = A^{*T} \cdot V \cdot A^* \tag{8.21}$$

8.1.1 Proprietăţile transformărilor unitare

În continuare vor fi prezentate câteva din proprietăţile transformărilor unitare.

- O transformare unitară conservă energia semnalului. Această proprietate o vom demonstra pentru cazul unei transformări unitare unidimensionale, pentru simplitate, deşi este perfect valabilă şi în cazul unei transformări bidimensionale. Fie $\underline{u}$ un semnal discret uni-dimensional, format din N eşantioane, şi o transformare unitară dată de matricea A. Relaţiile de transformare vor fi:

  $$\underline{v} = A \cdot \underline{u}$$
  $$\underline{u} = A^{*T} \cdot \underline{v}$$

  Energia semnalului $\underline{u}$ este dată de norma la pătrat a spaţiului în care este reprezentat semnalul:

  $$E_v = \|\underline{v}\|^2 = \underline{v}^{*T} \cdot \underline{v} = (A\underline{u})^{*T} \cdot A\underline{u} = \underline{u}^{*T} A^{*T} A\underline{u} = \underline{u}^{*T} \cdot \underline{u} = \|\underline{u}\|^2 = E_u \tag{8.22}$$

  În general transformarea se alege astfel încât energia să fie inegal distribuită în spaţiul transformatei, chiar dacă ea era uniform distribuită în spaţiul original.

- Entropia unui semnal discret cu valori aleatoare se conservă printr-o transformare unitară. Dar entropia este o măsură a cantităţii de informaţie, ceea ce înseamnă că o transformare unitară păstrează informaţia conţinută în semnal.

- Coeficienţii în spaţiul transformatei sunt decorelaţi sau aproape decorelaţi. Transformata optimă care compactează maximum de energie într-un număr dat de coeficienţi şi care în acelaşi timp decorelează complet aceşti coeficienţi, este transformarea Karhunen-Loève.

8.2 Transformata Fourier discretă

8.2.1 Transformata Fourier unidimensională

Pentru un semnal unidimensional, $\underline{u}$, de dimensiune N, de forma:

$$\underline{u} = [u(0), u(1), ..., u(N-1)]^T \tag{8.23}$$

transformarea Fourier directă este dată de relaţia:

$$v(k) = \sum_{n=0}^{N-1} u(n) \cdot e^{-\frac{2\pi jkn}{N}} \quad \vee\, k = 0..N-1 \tag{8.24}$$

iar transformarea Fourieri inversă de relaţia:

$$u(n) = \frac{1}{N} \sum_{k=0}^{N-1} v(k) \cdot e^{\frac{2\pi jkn}{N}} \quad \vee\, n = 0..N-1 \tag{8.25}$$

Astfel definită, transformarea Fourier nu este unitară. Următoarele relaţii definesc transformarea Fourier unitară, directă şi inversă:

$$v(k) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} u(n) \cdot e^{-\frac{2\pi jkn}{N}} \quad \vee\, k = 0..N-1 \tag{8.26}$$

$$u(n) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} v(k) \cdot e^{\frac{2\pi jkn}{N}} \quad \vee\, n = 0..N-1 \tag{8.27}$$

Dacă definim matricea $F = \{f(k,n)\}$ a transformării, având elementele:

$$f(k,n) = \frac{1}{\sqrt{N}} e^{-\frac{2\pi jkn}{N}} \quad k,n = 0..N-1 \tag{8.28}$$

atunci transformarea Fourier se poate scrie matricial astfel:

$$\underline{v} = F \cdot \underline{u} \tag{8.29}$$

$$\underline{u} = F^* \cdot \underline{v} \tag{8.30}$$

cu observaţia că matricea F are următoarea proprietate: $F = F^T$.

Pentru calculul transformatei Fourier discrete, există algoritmi rapizi (FFT[1]) care reduc complezitatea calculelor de la $O(N^2)$ la $O(NlogN)$.

8.2.2 Transformarea Fourier bidimensională

Pentru o imagine $U = \{u(m,n)\}_{m,n=0..N-1}$, de dimensiune $N \times N$, imaginea transformată $V = \{v(k,l)\}_{k,l=0..N-1}$ se calculează cu relaţia următoare, ce reprezintă transformarea Fourier în ipoteza separabilităţii:

$$v(k,l) = \frac{1}{N} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} u(m,m) \cdot e^{-\frac{2\pi j(km+ln)}{N}} \tag{8.31}$$

[1] Fast Fourier Transform

iar transformarea Fourier inversă este dată de formula:

$$u(m,n) = \frac{1}{N}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} v(k,l)\cdot e^{\frac{2\pi j(km+ln)}{N}} \tag{8.32}$$

Dacă folosim matricea F definită cu relaţia (8.28), atunci matricial se poate scrie:

$$V = F\cdot U\cdot F \tag{8.33}$$

$$U = F^{*}\cdot V\cdot F^{*} \tag{8.34}$$

8.3 Transformata cosinus discretă

Transformata cosinus este o transformată unitară separabilă, definită de matricea $C = \{c(k,n)\}$, ale cărei elemente sunt date de relaţia:

$$c(k,n) = \begin{cases} \frac{1}{\sqrt{N}}, & k = 0, 0 \le n \le N-1 \\ \sqrt{\frac{2}{N}} cos\frac{(2n+1)\pi k}{2N}, & 1 \le k \le N-1, 0 \le n \le N-1 \end{cases} \tag{8.35}$$

Transformata cosinus, directă şi inversă, pentru un semnal unidimensional, este dată de relaţiile:

$$v(k) = \alpha(k)\sum_{n=0}^{N-1} u(n) cos\frac{(2n+1)\pi k}{2N}, \quad 0 \le k \le N-1 \tag{8.36}$$

$$u(n) = \sum_{n=0}^{N-1} \alpha(k)v(k) cos\frac{(2n+1)\pi k}{2N}, \quad 0 \le n \le N-1 \tag{8.37}$$

unde

$$\alpha(0) = \sqrt{\frac{1}{N}}, \quad \alpha(k) = \sqrt{\frac{2}{N}}, \;\; 1 \le k \le N-1 \tag{8.38}$$

Transformarea cosinus bidimensională, directă şi inversă, este dată de următoarele două relaţii, scrise matricial:

$$V = C\cdot U\cdot C^{T} \tag{8.39}$$

$$U = C^{T}\cdot U\cdot C \tag{8.40}$$

deoarece matricea C are proprietatea că $C = C^{*}$, elementele sale fiind numere reale.

Observaţie: transformarea cosinus nu este partea reală a transformării Fourier.

8.4 Transformata sinus discretă

Transformata sinus este o transformată unitară separabilă, definită de matricea $S = \{s(k,n)\}$, ale cărei elemente sunt date de relaţia:

$$s(k,n) = \sqrt{\frac{2}{N+1}} sin \frac{(k+1)(n+1)\pi}{N+1}, \quad 0 \le k,n \le N-1 \qquad (8.41)$$

Relaţiile ce definesc transformarea sinus unidimensională, directă şi inversă, sunt următoarele:

$$v(k) = \sqrt{\frac{2}{N+1}} \sum_{n=0}^{N-1} u(n) sin \frac{(k+1)(n+1)\pi}{N+1}, \quad 0 \le k \le N-1 \qquad (8.42)$$

$$u(n) = \sqrt{\frac{2}{N+1}} \sum_{k=0}^{N-1} v(k) sin \frac{(k+1)(n+1)\pi}{N+1}, \quad 0 \le n \le N-1 \qquad (8.43)$$

Transformarea sinus bidimensională, directă şi inversă, se scrie matricial astfel:

$$V = S \cdot U \cdot S \qquad (8.44)$$

$$U = S \cdot V \cdot S \qquad (8.45)$$

deoarece matricea S are proprietatea că $S = S^* = S^T = S^{-1}$.

Observaţie: Transformarea sinus nu este partea imaginară a transformării Fourier.

DESFĂŞURAREA LUCRĂRII

Problema 1. Pentru o imagine de dimensiune $N \times N$, adică pătrată, observaţi imaginea transformată obţinută cu ajutorul transformatei cosinus bidimensională (funcţia `transformata_cosinus_discreta` din meniul `Algoritmi`). Codul acestei funcţii este prezentat în continuare:

```
void ImageViewer :: transformata_cosinus_discreta( void )
{
   int w, h;
   int i, j, k;
   double pi = 3.1415926;

   w = image.width();
```

```
 h = image.height();

 if( w == h )
 {
    int N = w;
    double max = 0;
    double C[ N ][ N ];
//matricea transformarii cosinus

int U[ N ][ N ];
//matricea imaginii in spatiul original

double V[ N ][ N ];
//matricea imaginii in spatiul transformatei

double  AUX[ N ][ N ];

    //formarea matricei C a transformarii cosinus discreta
    for( i = 0; i < N; i++ )
       C[ 0 ][ i ] = 1. / sqrt( N );

    for( i = 1; i < N; i++ )
       for( j = 0; j < N; j++ )
       {
          C[ i ][ j ] = sqrt( 2./N ) *
               cos( pi * ( 2*j + 1 ) * i / ( 2*N ) );

          if( C[ i ][ j ] > max )
             max = C[ i ][ j ];
     }

    //formarea matricei U
    for( i = 0; i < N; i++ )
       for( j = 0; j < N; j++ )
          U[ i ][ j ] = qRed( image.pixel( i, j ));

     //V = C*U*Ct
     //mai intii vom calcula AUX = C * U
     for( i = 0; i < N; i++ )
        for( j = 0; j < N; j++ )
        {
           AUX[ i ][ j ] = 0;
           for( k = 0; k < N; k++ )
           AUX[ i ][ j ] += C[ i ][ k ] * U[ k ][ j ];
```

```
         }

      //apoi V = AUX * Ct
      max = 0;
      for( i = 0; i < N; i++ )
         for( j = 0; j < N; j++ )
         {
            V[ i ][ j ] = 0;
            for( k = 0; k < N; k++ )
               V[ i ][ j ] += AUX[ i ][ k ] * C[ j ][ k ];

            if( V[ i ][ j ] > max )
               max = V[ i ][ j ];
         }

      QImage transf( N, N, 32, 0, QImage::IgnoreEndian );

      for( i = 0; i < N; i++ )
         for( j = 0; j < N; j++ )
         {
            int niv = (int)( V[ i ][ j ] * 255 / max );
            transf.setPixel( i, j, qRgb( niv, niv, niv ) );
         }

   image = transf;
   pm = image;
   update();
}
```

Problema 2. Implementaţi transformarea sinus bidimensională.

Problema 3. Implementaţi transformarea Fourier bidimensională (numai partea reală, care reprezintă spectrul de frecvenţe spaţiale al imaginii).

Problema 4. Verificaţi proprietatea de conservare a energiei pentru una din transformări.

Lucrarea 9

Compresia imaginilor

BREVIAR TEORETIC

Termenul de compresie se referă la totalitatea metodelor ce au drept scop reducerea cantităţii de date necesare pentru reprezentarea unei imagini. Compresia este folosită în special pentru stocarea sau transmiterea imaginilor.

Să considerăm cazul unei imagini de dimensiune 512×512 pixeli. Dacă aceasta este o imagine în tonuri de gri, iar fiecare pixel este codat cu 8 biţi, atunci cantitatea de date necesară pentru a reprezenta această imagine este:

$$512 \times 512 \times 8 = 2^9 \times 2^9 \times 2^3 = 2^{21} \approx 2\,Mb$$

Din acest calcul ne putem da seama că pentru a stoca o imagine avem nevoie de spaţiu considerabil, iar pentru transmiterea ei avem nevoie de un canal de transmisiune de bandă largă, de care nu dispunem întotdeauna.

9.1 Clasificarea metodelor de compresie

Metodele de compresie se pot clasifica astftel:

- **Metode de compresie la nivel de pixel**
  Aceste metode nu ţin cont de corelaţia care există între pixelii vecini, codând fiecare pixel ca atare. Acest tip de compresie este fără pierdere de informaţie, adică imaginea iniţială poate fi refăcută perfect din imaginea comprimată. Exemple de astfel de metode:
  - codarea Huffman
  - codarea LZW (Lempel-Ziv-Walsh)
  - codarea RLE (Run Length Encoding)
- **Metode de compresie predictive**
  Aceste metode realizează compresia folosind corelaţia care există între pixelii vecini, dintr-o imagine. Exemple de astfel de metode:

  - codarea cu modulaţie “delta”
  - codarea DPCM (Differential Pulse Code Modulation)

- **Metode de compresie cu transformate**
  Aceste metode se bazează pe scrierea imaginii într-o altă bază, prin aplicarea unei transformări unitare, atfel încât energia imaginii să fie concentrată într-un număr cât mai mic de coeficienţi.

- **Alte metode de compresie**
  - cuantizarea vectorială
  - codarea folosind fractali
  - codarea hibridă

9.2 Algoritmul Huffman

Să presupunem că valorile pixelilor unei imagini sunt simboluri ale unei surse S:

$$[S] = \{S_1, S_2, ..., S_N\} \tag{9.1}$$

pentru care se cunosc probabilităţile de apariţie:

$$[P] = \{p_1, p_2, ..., p_N\} \tag{9.2}$$

$$P(S_1) = p_1 \quad P(S_2) = p_2 \quad P(S_N) = p_N \tag{9.3}$$

Aceste probabilităţi nu reprezintă altceva decât frecveţele relative de apariţie ale simbolurilor într-un şir de simboluri emise de sursa S.

Entropia sursei S care generează simbolurile se calculează cu formula:

$$H(S) = -\sum_{i=1}^{N} p_i \cdot log p_i \tag{9.4}$$

Ne propunem să codăm simbolurile sursei S cu simboluri ale unei alte surse (de exemplu o sursă care generează doar două simboluri: 0 şi 1), astfel încât entropia noii surse să fie maximizată.

În continuare este prezentată o metodă care maximizează această entropie, metodă elaborată de Huffman în 1952:

Pasul 1. Se ordonează descrescător probabilităţile p_i.

Pasul 2. Se formează un arbore binar, având ca frunze valorile celor mai mici probabilităţi din şirul de probabilităţi. Rădăcina acestui arbore va conţine suma probabilităţilor celor două frunze ale sale. Se etichetează muchia stângă cu 1 şi muchia dreaptă cu 0.

Pasul 3. Din şirul P se elimină cele două probabilităţi care au fost alese ca fiind cele mai mici. În şirul P se introduce valoarea conţinută de rădăcina arborelui binar format.

Pasul 4. Dacă în şirul P există mai mult de un element, atunci se reia algoritmul, de la **Pasul 1**.

Pasul 5. Codarea binară a fiecărui element se obţine prin parcurgerea arborelui ce s-a format, de la rădăcină spre fiecare frunză.

Eficienţa codificării Huffman este dată de lungimea medie $\bar{l}$ a cuvintelor de cod, care se calculează folosind formula:

$$\bar{l} = \sum_{i=1}^{N} l_i \cdot p_i \tag{9.5}$$

unde l_i este lungimea codului alocat simbolului S_i.

Exemplu:
Fie o sursă S care generează 4 simboluri, $[S] = \{a, b, c, d\}$, care au următoarele probabilităţi de apariţie: $[P] = \{0.2; 0.4; 0.1; 0.3\}$. Arborele codării Huffman se construieşte conform etapelor prezentate în Figurile 9.1, 9.2, 9.3 şi 9.4.

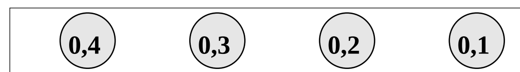


Figura 9.1: Algoritmul Huffman: etapa 1.

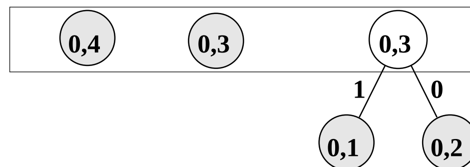


Figura 9.2: Algoritmul Huffman: etapa 2.

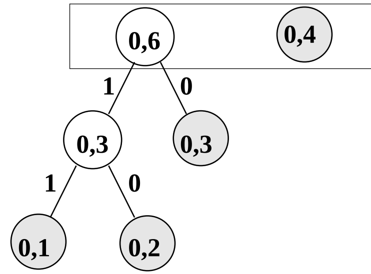


Figura 9.3: Algoritmul Huffman: etapa 3.

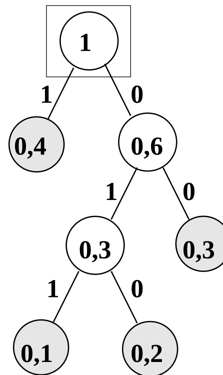


Figura 9.4: Algoritmul Huffman: etapa 4.

Pentru decompresie este necesară o tabelă în care să se memoreze corespondenţele între simboluri şi cuvintele de cod. Fără aceasta decompresia este imposibil de realizat.

Simbol	**Cuvânt de cod**
a	"010"
b	"1"
c	"011"
d	"00"

Lungimea medie a cuvintelor de cod, pentru acest exemplu, este:

$$\bar{l} = \sum_{i=1}^{4} p_i \cdot l_i = 0,2 \cdot 3 + 0,4 \cdot 1 + 0,1 \cdot 3 + 0,3 \cdot 2 = 1,9\, bits/simbol$$

Dacă nu am fi codat simbolurile, în vederea maximizării entropiei sursei, ar fi fost nevoie de 2 biţi/simbol pentru codare.

Pentru imagini, probabilităţile de apariţie ale nivelelor de gri se obţin prin calcularea histogramei imaginii. Dacă histograma este uniformă, atunci algoritmul Huffman de codare nu este eficient, nerealizând nici o îmbunătăţire a lungimii cuvintelor de cod.

9.3 Algoritmul RLE

9.3.1 Algoritmul RLE pentru imagini binare

Vom considera valorile pixelilor (0 sau 255) ca fiind simbolurile 0 şi 1 generate de o sursă binară. În vederea codării imaginea este transformată într-un şir unidimensional, prin concatenarea liniilor sau a coloanelor, ca în Figura 9.5.

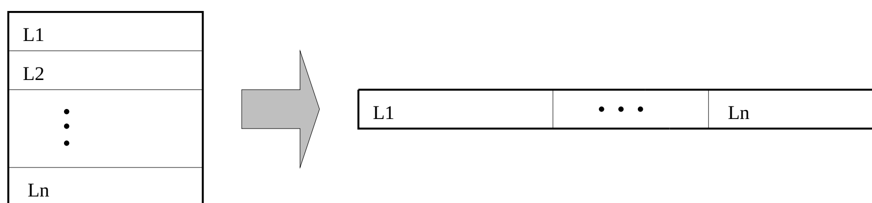


Figura 9.5: Transformarea imaginii într-un şir unidimensional, prin concatenarea liniilor.

Acest şir de elemente 0 şi 1 va fi codat, codarea realizându-se astfel: primul element al şirului codat este primul element din şirul de codat; apoi se scrie în şirul codat lungimea fiecărui subşir constant din şirul de codat.

Exemplu:
şirul de codat: 0000000111110001100000000010100011111111111
şirul codat: 0 7 5 3 2 9 1 1 1 3 12

Acest tip de codare se foloseşte în special pentru comprimarea imaginilor transmise prin fax.

Decompresia se face similar cu compresia, parcurgând şirul codat şi generând şiruri alternate, de simboluri 0 sau 1, începând cu primul element din şirul codat, şi de lungimi indicate de valorile întâlnite în şirul de decodat.

9.3.2 Algoritmul RLE pentru imagini în tonuri de gri

Pentru imagini în tonuri de gri, algoritmul RLE se aplică pentru plane formate din biţii de pe aceeaşi poziţie, din reprezentarea binară a valorilor pixelilor. De exemplu, dacă imaginea în tonuri de gri, are 256 de nivele de gri, corespunzător la o cuantizare pe 8 biţi, atunci din această imagine se

construiesc 8 plane (sau 8 imagini binare) astfel: o imagine binară formată din biţii b_0, o altă imagine binară din biţii b_1, ş.a.m.d. (vezi Figura 9.6).

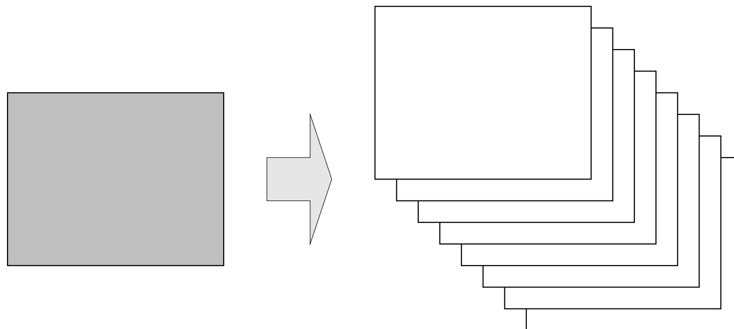


Figura 9.6: Transformarea unei imagini cu 256 nivele de gri, în 8 imagini binare.

Valoarea pixelului (i, j) din imaginea în tonuri de gri va fi reprezentată pe 8 biţi astfel:

$$val(i, j) = [b_0b_1b_2b_3b_4b_5b_6b_7]$$

unde b_0 este cel mai semnificativ bit (MSB[1]), iar b_7 este cel mai puţin semnificativ bit (LSB[2]).

Imaginea binară formată din biţii cei mai semnificativi va fi comprimată cel mai bine cu algoritmul RLE. Imaginea binară formată din biţii cei mai puţin semnificativi va fi o imagine cu "purici", pentru care se poate lua decizia de a nu mai fi codată şi deci ignorată.

9.4 Compresia cu transformate

Compresia cu ajutorul transformatelor se bazează pe proprietatea acestora de a compacta energia imaginii într-un număr redus de coeficienţi, cât mai decorelaţi, repartizaţi neuniform în spaţiul transformării.

Formula care defineşte o transformarea directă este următoarea:

$$V = A \cdot U \cdot A^T \tag{9.6}$$

unde A este matricea ce defineşte o transformare unitară, separabilă.

Pentru compresia imaginilor, transformarea cea mai apropiată din punct de vedere al performanţelor de transformarea optimă Karhunen-Loéve, este transformarea cosinus. Coeficienţii de energie mare sunt situaţi în colţul

[1]Most Significant Bit.
[2]Least Significant Bit.

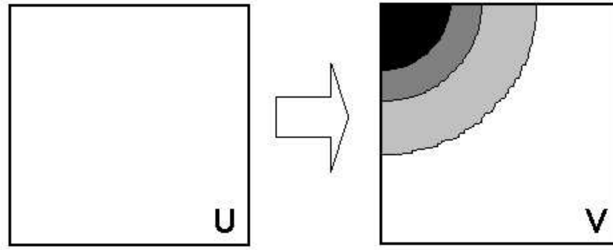


Figura 9.7: Transformarea directă.

din stânga-sus al imaginii transformate, în cazul în care se foloseşte pentru compresie transformarea cosinus (vezi Figura 9.7).

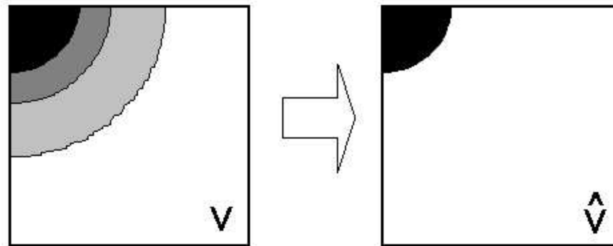


Figura 9.8: Anularea coeficienţilor de energie mică.

Pentru a obţine o rată de compresie mai mare, vor fi anulaţi coeficienţii de energie mică (vezi Figura 9.8). Anularea acestor coeficienţi va duce, însă, la scăderea calităţii imaginii după decompresie. Adică, prin transformare inversă (vezi Figura 9.9), imaginea obţinută din imaginea $\hat{V}$, nu va fi exact imaginea originală U.

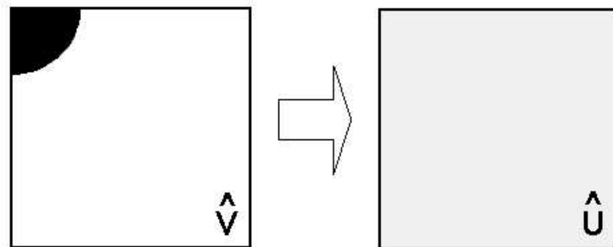


Figura 9.9: Transformarea inversă.

$$\hat{U} = A^{*T} \cdot \hat{V} \cdot A^{*} \tag{9.7}$$

Compresia cu transformarea cosinus stă la baza algoritmului JPEG[3] de compresie a imaginilor.

DESFĂŞURAREA LUCRĂRII

Problema 1. Implementaţi în C unul dintre algoritmii prezentaţi (Huffman sau RLE). Pentru simplitate, imaginea citită în format BMP, va fi scrisă într-un format simplificat (vezi funcţia `write_naked_image`, din meniul `Algortimi`). Imaginea comprimată va fi scrisă într-un fişier cu exensia `.huf` sau `.rle`.

```
void ImageViewer :: write_naked_image( void )
{
   int i, j;
   int w, h;
   FILE *file;

   w = image.width();
   h = image.height();

   file = fopen( "naked.img", "w" );
   if( file != NULL )
   {
      for( i = 0; i < w; i++ )
      {
         for( j = 0; j < h; j++ )
         {
            int niv = qRed( image.pixel( i, j ) );
            //nivelul de gri al pixelului

            fprintf( file, "%3d ", niv );
         }
         fprintf( file, "\n" );
      }

      fclose( file );
   }
}
```

Problema 2. Calculaţi raportul de compresie obţinut, ca raport dintre dimensiunile celor două fişiere: cel original (`naked.img`) în formatul simplificat şi cel comprimat.

[3]Joint Photographic Experts Group.

Problema 3. Observaţi efectul suprimării coeficienţilor de energie joasă, la o compresie-decompresie folosind transformata cosinus discretă. (funcţia `compresie_decompresie_cu_DCT`, din meniul `Algoritmi`). Pentru aceasta vizualizaţi imaginea `decompresata.bmp`. Codul prezentat în continuare presupune că imaginea este pătrată:

```
void ImageViewer :: compresie_decompresie_cu_DCT( void )
{
   int w, h;
   int i, j, k;
   double pi = 3.1415926;

   w = image.width();
   h = image.height();

   int N = w;
   double  max;
   double  C[ N ][ N ]; //matricea transformarii cosinus
   double  U[ N ][ N ]; //imaginea in spatiul original
   double  V[ N ][ N ]; //imaginea in spatiul transformatei
   double  AUX[ N ][ N ];

   // COMPRESIA IMAGINII
   //formarea matricei C a transformarii cosinus discreta
   for( i = 0; i < N; i++ )
      C[ 0 ][ i ] = 1. / sqrt( N );

   for( i = 1; i < N; i++ )
      for( j = 0; j < N; j++ )
         C[ i ][ j ] = sqrt( 2./N ) *
            cos( pi * ( 2*j + 1 ) * i / ( 2*N ) );

   //formarea matricei U
   for( i = 0; i < N; i++ )
      for( j = 0; j < N; j++ )
         U[ i ][ j ] = qRed( image.pixel( i, j ));

   //V = C*U*Ct
   //mai intii vom calcula AUX = C * U
   for( i = 0; i < N; i++ )
      for( j = 0; j < N; j++ )
      {
         AUX[ i ][ j ] = 0;
         for( k = 0; k < N; k++ )
```

```
            AUX[ i ][ j ] += C[ i ][ k ] * U[ k ][ j ];
      }

   //apoi V = AUX * Ct
   max = 0;
   for( i = 0; i < N; i++ )
      for( j = 0; j < N; j++ )
      {
         V[ i ][ j ] = 0;
         for( k = 0; k < N; k++ )
            V[ i ][ j ] += AUX[ i ][ k ] * C[ j ][ k ];

         if( V[ i ][ j ] > max )
            max = V[ i ][ j ];
      }

   //anularea coeficientilor
   //in vederea maririi factorului de compresie
   for( i = 0; i < N; i++ )
      for( j = 0; j < N; j++ )
      {
         if( V[ i ][ j ] < 100 )  //pragul de anulare
            V[ i ][ j ] = 0;

         //alte valori prag: -500, -100, 0, 100, 500
      }

   // DECOMPRESIA IMAGINII
   //U = Ct * V * C
   //AUX = Ct * V
   for( i = 0; i < N; i++ )
      for( j = 0; j < N; j++ )
      {
         AUX[ i ][ j ] = 0;
         for( k = 0; k < N; k++ )
            AUX[ i ][ j ] += C[ k ][ i ] * V[ k ][ j ];
      }

   //apoi U = AUX * C
   for( i = 0; i < N; i++ )
      for( j = 0; j < N; j++ )
      {
         U[ i ][ j ] = 0;
         for( k = 0; k < N; k++ )
```

```
            U[ i ][ j ] += AUX[ i ][ k ] * C[ k ][ j ];
      }

   //pseudo-imaginea diferenta
   QImage diff( N, N, 32, 0, QImage:: IgnoreEndian );

   for( i = 0; i < N; i++ )
      for( j = 0; j < N; j++ )
      {
         int dif = abs( qRed( image.pixel( i, j ) ) -
            (int)( U[ i ][ j ] ) );
         diff.setPixel( i, j, qRgb( dif, dif, dif ) );
      }

   iio.setImage( diff );
   iio.setFileName( "diferenta.bmp" );
   iio.setFormat( "BMP" );
   iio.write();
}
```

Problema 4. Calculaţi eroarea pătratică medie (ε) dintre imaginea originală şi imaginea obţinută prin compresia şi decompresia cu transformată cosinus discretă, folosind formula:

$$\varepsilon = \overline{(U-V)^2} = \frac{1}{NM}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}[u(i,j) - v(i,j)]^2$$

Pentru aceasta modificaţi funcţia `Compresie-decompresie cu DCT`. Observaţi valorile erorii medii pătratice pentru diverse valori ale pragului de anulare a coeficienţilor în spaţiul transformării.

Problema 5. Observaţi pseudo-imaginea diferenţă, dintre imaginea originală şi cea obţinută prin compresia şi decompresia cu DCT, pentru diferite valori ale pragului (vezi fişierul `diferenta.bmp`).

Lucrarea 10

Segmentarea imaginilor

BREVIAR TEORETIC

Segmentarea reprezintă împărţirea imaginii în zone de interes, după anumite criterii. Fiecărui pixel i se va atribui o valoare, 0 sau 1, reprezentând apartenenţa acestuia la o anumită zonă sau regiune de interes. De regulă, segmentarea urmăreşte extragerea, identificarea sau recunoaşterea unui anumit obiect dintr-o imagine. Zonele sau regiunile care alcătuiesc o imagine poartă numele de segmente. Pentru o imagine $f(m,n)$, segmentarea reprezintă împărţirea lui f într-un număr N de zone $f_i(m,n)$, cu $i = 1..N$, ca în Figura 10.1. Aceste segmente se numesc complete, dacă au următoarele proprietăţi:

- $f_i \bigcap f_j = \emptyset \;\; pentru \; i \neq j$,
- $\bigcup\limits_{i=1}^{N} f_i = f$,
- segmentul f_i să fie compact, pentru $\vee i$,
- pentru $\vee i$, un anumit criteriu de uniformitate $E(f_i)$ este satisfăcut,
- pentru $\vee i, j$, citeriul de uniformitate pentru $f_i \bigcup f_j$ nu este satisfăcut.

Metodele de segmentare a imaginilor se pot clasifica în:

- metode de segmentare orientate pe regiuni
- metode de segmentare orientate pe contururi

10.1 Segmentarea orientată pe regiuni

În general, operaţia de segmentare orientată pe regiuni urmăreşte extragerea din imagine a zonelor (regiunilor) ocupate de diversele obiecte prezente

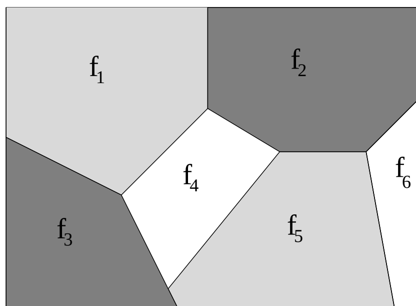


Figura 10.1: Exemplu teoretic de segmentare.

în scenă. Un obiect se defineşte ca o entitate caracterizată de un set de parametri ale căror valori nu se modifică în diferitele puncte ce aparţin entităţii considerate. Unul dintre cei mai simpli parametri de definiţie este nivelul de gri al pixelului. Dacă nivelul de gri caracterizează în mod suficient obiectele din imagine, atunci histograma imaginii va prezenta o structură de moduri dominante - adică de intervale de nivele de gri ce apar cu probabilitate mai mare. Fiecare mod al histogramei va reprezenta câte un obiect sau o categorie de obiecte.

10.1.1 Prăguirea histogramei

Separarea modurilor histogramei, şi deci identificarea obiectelor din imagine, se face prin alegerea unor nivele de gri, numite praguri de segmentare. De obicei aceste praguri se aleg ca fiind corespunzătoare minimelor locale ale histogramei. Din imaginea iniţială f se construieşte o imagine de etichete g (imagine etichetată), conform unei transformări de forma:

$$g(m,n) = \begin{cases} E_0, & 0 \leq f(m,n) < T_{K-1} \\ E_K, & T_{K-1} \leq f(m,n) < L \end{cases} \tag{10.1}$$

În cazul unei histograme bimodale (care conţine două moduri dominante), ca cea din Figura 10.2(a), transformarea de mai sus devine:

$$g(m,n) = \begin{cases} E_0, & 0 \leq f(m,n) < T \\ E_1, & T \leq f(m,n) < L \end{cases} \tag{10.2}$$

asemănătoare cu operaţia de binarizare. O astfel de histogramă bimodală este caracteristică imaginilor ce conţin un singur obiect sau mai multe obiecte de acelaşi fel, pe un fundal uniform. De exemplu, o imagine care conţine un scris negru pe un fond alb.

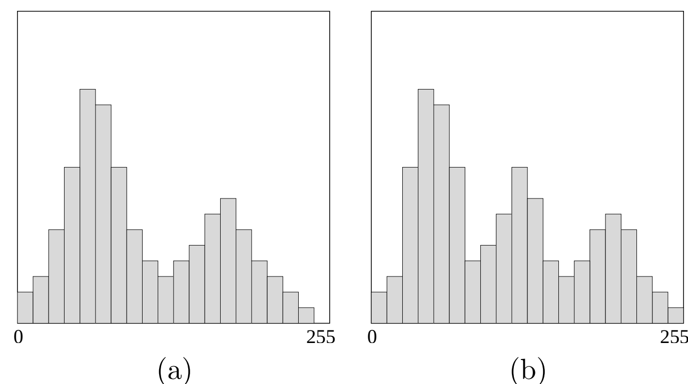


Figura 10.2: Histogramă: (a) bimodală; (b) cu trei moduri dominante.

10.1.2 Segmentarea prin creştere de regiuni

Principiul pe care se bazează creşterea regiunilor este următorul: se aleg în imagine pixeli reprezentativi pentru fiecare obiect individual, pe baza cărora are loc un proces de aglomerare a pixelilor vecini acestora, ce au aceleaşi proprietăţi cu pixelii reprezentativi. În urma acestui proces de aglomerare se obţin zone de pixeli cu aceleaşi caracteristici.

Procesul se opreşte în momentul în care fiecare pixel al imaginii a fost alocat unei regiuni. Metoda are două etape esenţiale: alegerea punctelor de start (puncte iniţiale), numite *germeni* sau *seminţe*, şi creşterea propriu-zisă a regiunilor. Numărul final de regiuni rezultate este egal cu numărul de germeni aleşi iniţial pentru creştere. În principiu, este de dorit ca fiecare obiect individual aflat în imagine, să fie marcat de un germene. Daca în interiorul unui obiect se găsesc mai mulţi germeni, pentru fiecare dintre ei va fi crescută o regiune, fapt ce duce la o segmentare artificială nedorită a obiectului respectiv. Acest neajuns poate fi corectat printr-o etapă de fuziune a regiunilor adiacente ce au proprietăţi asemănătoare.

Daca în interiorul unui obiect nu este ales nici un germene, atunci obiectul respectiv va fi inclus în regiunile ce cresc pornind de la germeni din vecinătatea sa.

10.2 Segmentarea orientată pe contururi

Într-o imagine, variaţiile de nivel ale pixelilor reprezintă schimbări ale proprietăţilor fizice sau geometrice ale obiectelor ce compun scena. Într-un număr mare de cazuri, aceste variaţii de intensitate corespund frontierelor (contururilor) regiunilor determinate de obiectele dintr-o imagine.

10.2.1 Tehnicile de gradient

Principiul acestor metode constă în definirea punctelor de contur ca fiind acei pixeli ai imaginii pentru care apar schimbări abrupte ale nivelului de gri. Măsurarea acestor variaţii se face prin operatori derivativi de tip gradient. Derivata imaginii pe direcţia r, ce face unghiul θ cu orizontala, este dată de combinaţia liniară a derivatelor parţiale pe direcţiile orizontală şi verticală:

$$\frac{\partial f}{\partial r} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial r} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial r} = \frac{\partial f}{\partial x} cos\theta + \frac{\partial f}{\partial y} sin\theta \tag{10.3}$$

$$\frac{\partial f}{\partial r} = f_x cos\theta + f_y sin\theta \tag{10.4}$$

Valoarea maximă a acestei derivate, calculate după unghiul θ este determinată de ecuaţia:

$$\frac{\partial}{\partial \theta}\left(\frac{\partial f}{\partial r}\right) = -f_x sin\theta + f_y cos\theta = 0 \tag{10.5}$$

care are soluţia:

$$\theta = arctan\left(\frac{f_x}{f_y}\right) \tag{10.6}$$

Pe această direcţie, modulul gradientului este:

$$\left(\frac{\partial f}{\partial r}\right)_{max} = \sqrt{f_x^2 + f_y^2} \tag{10.7}$$

Din punct de vedere practic, implementarea acestei metode implică calcularea, pentru fiecare pixel al imaginii, a derivatelor parţiale f_x şi f_y, calcularea modulului gradientului maxim şi a direcţiei acestuia. Valoarea gradientului maxim din fiecare pixel al imaginii este apoi comparată o valoare de prag: dacă pragul este depăşit, atunci pixelul este considerat a fi pixel de contur. Realizarea derivatelor parţiale după direcţiile orizontală şi verticală implică formularea discretă a lui f_x şi f_y:

$$f_x = \frac{\partial f}{\partial x} = \frac{\Delta f(m,n)}{\Delta m} \tag{10.8}$$

$$f_y = \frac{\partial f}{\partial y} = \frac{\Delta f(m,n)}{\Delta n} \tag{10.9}$$

Aceste derivate parţiale discrete pot avea mai multe implementări:

$$f_x = f(m,n) - f(m+1,n) \quad f_y = f(m,n) - f(m,n+1) \tag{10.10}$$

$$f_x = f(m-1,n) - f(m,n) \quad f_y = f(m,n-1) - f(m,n) \tag{10.11}$$

$$f_x = f(m-1,n) - f(m+1,n) \quad f_y = f(m,n-1) - f(m,n+1) \tag{10.12}$$

Aceste expresii nu reprezintă altceva decât combinaţii liniare ale valorilor unor pixeli din imagine, situaţi în vecinătatea pixelului curent din poziţia (m,m). Prin urmare, se pot implementa folosind următoarele măşti de filtrare:

$$W_x = \begin{pmatrix} 1^* & -1 \end{pmatrix} \quad W_y = \begin{pmatrix} 1^* \\ -1 \end{pmatrix}$$

$$W_x = \begin{pmatrix} 1 & -1^* \end{pmatrix} \quad W_y = \begin{pmatrix} 1 \\ -1^* \end{pmatrix}$$

$$W_x = \begin{pmatrix} 1 & 0^* & -1 \end{pmatrix} \quad W_y = \begin{pmatrix} 1 \\ 0^* \\ -1 \end{pmatrix}$$

unde prin * am marcat originea măştii de filtrare. O simplificare uzuală practică este dată de aproximarea:

$$\left(\frac{\partial f}{\partial r}\right)_{max} \approx \mid f_x \mid + \mid f_y \mid \tag{10.13}$$

Folosirea măştilor de derivare pe verticală şi orizontală prezentate are însă serioase neajunsuri: dimensiunea lor mică face ca rezultatele să fie extrem de sensibile la zgomote. În aceste condiţii a apărut ideea naturală de a combina filtrarea de derivare cu o filtrare de netezire, care să reducă efectele zgomotului. Ceea ce rezultă pentru operatorii de derivare orizontală şi verticală sunt măştile:

$$W_x = \begin{pmatrix} 1 & 0 & -1 \\ c & 0^* & -c \\ 1 & 0 & -1 \end{pmatrix} \quad W_y = \begin{pmatrix} 1 & c & 1 \\ 0 & 0^* & 0 \\ -1 & -c & -1 \end{pmatrix} \tag{10.14}$$

Prin particularizarea valorilor constantei c se pot obtine diverse tipuri de operatori de extragere de contur clasici: Prewitt ($c = 1$), Izotrop ($c = \sqrt{2}$) sau Sobel ($c = 2$). În Figura 10.3 puteţi observa efectele extragerii de contur folosind operatorul Sobel.



(a) (b)

Figura 10.3: Filtrul Sobel: (a) imaginea originală; (b) imaginea rezultată.

10.2.2 Operatorii compas

Un operator compas este definit de un număr de măşti de derivare, corespunzătoare unor filtrări liniare, pe direcţiile principale (verticală, orizontală şi cele două diagonale), în cele două sensuri. Compasul clasic are $D = 8$ măşti de filtrare, fiecare dintre ele realizând o derivare după o direcţie multiplu de 45°. Un exemplu de măşti de derivare direcţională sunt măştile următoare, indexate după direcţia geografică pe care se calculează derivata:

$$W_N = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \quad W_{NV} = \begin{pmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

$$W_V = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \quad W_{SV} = \begin{pmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{pmatrix}$$

$$W_S = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} \quad W_{SE} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -1 \end{pmatrix}$$

$$W_E = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix} \quad W_{NE} = \begin{pmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{pmatrix}$$

Dupa ce se vor calcula, pentru fiecare pixel în parte, cele opt valori ale gradienţilor, corespunzatoare celor opt măşti de derivare:

$$f_1(m,n) \quad f_2(m,n) \quad ... \quad f_8(m,n) \tag{10.15}$$

se va determina valoarea maximă dintre aceste opt valori. Această valoare va fi comparată cu valoarea de prag, iar în cazul în care este mai mare, pixelul respectiv va fi considerat pixel de contur.

10.2.3 Identificarea trecerilor prin zero ale celei de-a doua derivate

Unul dintre principalele dezavantaje ale metodelor de gradient este precizia slabă de localizare a conturului în condiţiile unei pante puţin abrupte (tranziţii slabe, graduale) a acestuia. Derivata a doua poate fi însă folosită pentru a marca centrul tranziţiei (trecerea sa prin zero). Operatorul bazat pe trecerea prin zero a derivatei secunde este operatorul "zero-crossing". În cazul imaginilor, derivata secundă trebuie luată în considerare după ambele direcţii, combinate în laplacian:

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{10.16}$$

În continuare sunt prezentate trei măşti ce implementează o derivată secundă bidirecţională (operator Laplace):

$$\begin{pmatrix} 0 & -\frac{1}{4} & 0 \\ -\frac{1}{4} & 1 & \frac{1}{4} \\ 0 & \frac{1}{4} & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{4} & -\frac{1}{2} & \frac{1}{4} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ \frac{1}{4} & -\frac{1}{2} & \frac{1}{4} \end{pmatrix} \begin{pmatrix} -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} \\ -\frac{1}{8} & 1 & -\frac{1}{8} \\ -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} \end{pmatrix}$$

Pentru fiecare pixel din imagine se va calcula derivata a doua şi dacă aceasta este zero, atunci pixelul este considerat ca fiind pixel de contur.

DESFĂŞURAREA LUCRĂRII

Problema 1. Implementaţi segmentarea prin praguirea histogramei. Pentru aceasta veti folosi codul de la lucrarea nr. 4 (egalizarea histogramei) şi imaginea `tools.bmp`.

Problema 2. Observaţi extragerea contururilor prin tehnici de gradient, folosind un operator Prewitt.

```
void ImageViewer :: extragere_contururi( void )
{
   int i, j;
   int k, l;
   int w, h;
```

```
   int g, g1, g2;
   int h1[ 3 ][ 3 ], h2[ 3 ][ 3 ];

   //operatori Prewitt
   h1[0][0] = -1;  h1[0][1] = 0;  h1[0][2] = 1;
   h1[1][0] = -1;  h1[1][1] = 0;  h1[1][2] = 1;
   h1[2][0] = -1;  h1[2][1] = 0;  h1[2][2] = 1;

   h2[0][0] = -1;  h2[0][1] = -1; h2[0][2] = -1;
   h2[1][0] = 0;   h2[1][1] = 0;  h2[1][2] = 0;
   h2[2][0] = 1;   h2[2][1] = 1;  h2[2][2] = 1;

   w = image.width();
   h = image.height();

   QImage image_ext( w, h, 32, 0, QImage::IgnoreEndian );

   for( i = 1; i < w - 1; i++ )
      for( j = 1; j < h - 1; j++ )
      {
         g1 = 0;
         g2 = 0;

         for( k = -1; k < 2; k++ )
            for( l = -1; l < 2; l++ )
            {
               g1 += h1[ k + 1 ][ l + 1 ]
                       * qRed( image.pixel( i+k, j+l ));
               g2 += h2[ k + 1 ][ l + 1 ]
                       * qRed( image.pixel( i+k, j+l ));
            }

          g = abs( g1 ) + abs( g2 );

          if( g > 100 )
             image_ext.setPixel( i, j, qRgb(255,255,255) );
          else
             image_ext.setPixel( i, j, qRgb(0,0,0) );
   }

   image = image_ext;
   pm = image;
   update();
}
```

Problema 3. Implementaţi operatorul Izotrop de extragere a conturarilor.

Problema 4. Implementaţi operatorul Sobel de extragere a conturarilor.

Problema 5. Implementaţi operatorul compas dat drept exemplu.

Problema 6. Implementaţi extragerea de contururi prin identificarea trecerilor prin zero ale derivatei secunde.

Problema 7. Implementaţi creşterea de regiuni, pornind de la un singur germene. Veţi folosi pentru aceasta imaginea `tools.bmp`.

Lista figurilor

2.1 Formarea imaginii în aparatul de fotografiat: 1-lentilă, 2-camera obscură, 3-obiectiv, 4-peliculă. 6
2.2 Forma tipică a funcţiei de eficienţă luminoasă relativă. 7
2.3 Domeniu finit din R^2. 8
2.4 Reţea dreptunghiulară de eşantionare. 9
2.5 Exemplu de funţie de cuantizare. 10

3.1 Reprezentarea unei operaţii punctuale. 16
3.2 Funcţia de accentuare de contrast. 16
3.3 Funcţia de întindere maximă a contrastului. 17
3.4 Întinderea maximă a contrastului: (a) imaginea originală şi (b) imaginea rezultată, pentru a=50, b=200. 18
3.5 Funcţia de binarizare. 19
3.6 Binarizarea: (a) imaginea originală şi (b) imaginea binarizată cu T=127. 19
3.7 Funcţia de negativare. 20
3.8 Negativarea: (a) imaginea originală şi (b) negativul imaginii. 20
3.9 Funcţia de decupare cu păstrarea fundalului. 21
3.10 Funcţia de decupare fără păstrarea fundalului. 21

4.1 Egalizarea histogramei: (a) imaginea originală şi (b) imaginea rezultată. 28
4.2 Histograma (a) originală şi (b) după egalizare. 28

5.1 Translatarea unui obiect dreptunghiular. 32
5.2 Translatarea spre dreapta a unui obiect dreptunghiular într-o imagine. 32
5.3 Rotaţia. 33
5.4 Oglindirea (a) “stânga-dreapta” (b) “sus-jos”. 33
5.5 Oglindirea “stânga-dreapta”: (a) imaginea originală; (b) imaginea oglindită. 34

6.1 Modelul aditiv de degradare. 38

6.2 Funcţia de densitate de probabilitate pentru o distribuţie uniformă. 39
6.3 Zgomotul uniform: (a) imaginea originală; (b) imaginea afectată de zgomot uniform, SNR=5 dB. 39
6.4 Funcţia de densitate de probabilitate pentru o distribuţie gaussiană. 40
6.5 Zgomotul Gaussian: (a) imaginea originală; (b) imaginea afectată de zgomot gaussian, SNR=5 dB. 40
6.6 Zgomotul "sare şi piper": (a) imaginea originală; (b) imaginea cu 10% pixeli afectaţi de zgomot. 41
6.7 Diverse funcţii de distribuţie. 42

7.1 Mască de filtrare pătrată de dimensiune 3x3. 48
7.2 Mască de filtrare pătrată de dimensiune 3×3. 49
7.3 Mască de filtrare pătrată de dimensiune 5×5. 50
7.4 Mască de filtrare pătrată de dimensiune 7×7. 50
7.5 Filtrarea de mediere: (a) imaginea originală; (b) imaginea filtrată cu o mască 7x7. 50
7.6 Filtrarea Laplace: (a) imaginea originală; (b) imaginea rezultată (negativată). 52
7.7 Filtrul de accentuare. 53
7.8 Filtrul median: (a) imaginea originală afectată de zgomot "sare şi piper"; (b) imaginea filtrată. 54

9.1 Algoritmul Huffman: etapa 1. 71
9.2 Algoritmul Huffman: etapa 2. 71
9.3 Algoritmul Huffman: etapa 3. 72
9.4 Algoritmul Huffman: etapa 4. 72
9.5 Transformarea imaginii într-un şir unidimensional, prin concatenarea liniilor. 73
9.6 Transformarea unei imagini cu 256 nivele de gri, în 8 imagini binare. 74
9.7 Transformarea directă. 75
9.8 Anularea coeficienţilor de energie mică. 75
9.9 Transformarea inversă. 75

10.1 Exemplu teoretic de segmentare. 82
10.2 Histogramă: (a) bimodală; (b) cu trei moduri dominante. . . . 83
10.3 Filtrul Sobel: (a) imaginea originală; (b) imaginea rezultată. . . 86

Bibliografie

[1] **V. Buzuloiu**, *"Prelucrarea Imaginilor"*, note de curs.

[2] **C. Vertan**, *"Prelucrarea şi Analiza Imaginilor"*, Ed. Printech, 1999.

[3] **A. Jain**, *"Fundamentals of Digital Image Processing"*, Prentice-Hall, 1989.

[4] **R. Gonzales, R. Woods**, *"Digital Image Processing"*, Addison Wesley, 1992.

[5] **Trolltech**, *"Qt Overview"*, http://www.trolltech.com/

[6] **Trolltech**, *"The Qt Class Reference"*, http://www.trolltech.com/

ISBN (10) 973-635-674-4
ISBN (13) 978-973-635-674-2